ADVANCED DECISION SYSTEMS

ADS TR-3154-01

PERFORMANCE EVALUATION OF ARTIFICIAL INTELLIGENCE SYSTEMS

Prepared by:

Richard M. Tong
Neville D. Newman
Gary Berg-Cross
Fred Rook

August 17, 1987

Final Technical Report for Period:
26 September 1986 - 30 June 1987

Contract Number: DAAH01-86-C-1053
Effective Date of Contract: September 26, 1986
Contract Expiration Date: June 30, 1987
ADS Project Number: 3154

Approved for Public Release; Distribution Unlimited

Sponsored by:

Defense Advanced Research Projects Agency (DoD)
Defense Small Business Innovation Research Program
ARPA Order No. 5916
Issued by U.S. Army Missile Command

201 San Antonio Circle, Suite 286
Mountain View, California 94040-1289
415 941-3912 FAX: 415 949-4029

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A184 054

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ADS TR-3154-01	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Advanced Decision Systems 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION U.S. Army Missile Command
6c. ADDRESS (City, State and ZIP Code) 201 San Antonio Circle, Suite 286 Mountain View, CA 94040-1289	7b. ADDRESS (City, State and ZIP Code) Redstone Arsenal Alabama 35898-5244
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Defense Advanced Research Projects Agency 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAAH01-86-C-1053
8c. ADDRESS (City, State and ZIP Code) 1400 Wilson Boulevard Arlington, VA 22209-2308	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) Performance Evaluation of Artificial Intelligence Systems	
12. PERSONAL AUTHOR(S) Tong, Richard M.; Newman, Neville D.; Berg-Cross, Gary; Rook, Fred	
13a. TYPE OF REPORT Final	13b. TIME COVERED FROM 86/09/26 TO 87/06/30
14. DATE OF REPORT (Yr., Mo., Day) 87/08/17	15. PAGE COUNT 124
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
09	02	
14	02	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Performance Evaluation, Knowledge-Based Systems, Expert Systems, Integrated Evaluation Methodology

19. ABSTRACT (Continue on reverse if necessary and identify by block number) The major premise of this effort is that the DoD is moving towards the large-scale use of Artificial Intelligence Systems (AISs) as tools for supporting decision making and information analysis in military domains. It becomes increasingly important, therefore, that we develop mechanisms both for assessing the impact that these systems have upon the ability of the individual decision makers and analysts to carry out their assignments, and for assessing the organizational impact that this would have. This is what we choose to call the problem of Performance Evaluation. ~~In~~ this report, ~~we~~ describe the development of a methodology for performance evaluation. ~~In our~~ approach, ~~we~~ provide for both multiple perspectives on the problem and multiple evaluation techniques. The central idea in our methodology is that of an evaluation frame. This is the main organizational element in ~~our~~ representation of performance knowledge. To validate the methodology, ~~we performed~~ a number of example evaluations of expert sytems building tools and decision aids, also using the results to help us define a prototype evaluation environment. The evaluations and the design are reported in full. We conclude ~~that our~~ proposed methodology has a high likelihood of success.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☒ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Allen Sears	22b. TELEPHONE NUMBER (Include Area Code) (202) 694-5921 22c. OFFICE SYMBOL

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

This report contains a description of the work performed under Government Contract Number DAAH01-86-C-1053, "Performance Evaluation of Artificial Intelligence Systems," during the period 26 September 1986 to 30 June 1987.

The report contains a discussion of efforts in all four tasks in the original statement of work. Our work on Task 1, "Development of a Methodology for Evaluating the Performance of Artificial Intelligence Systems," is described in Chapter 2. Work on Task 2, "Design of a Prototype Evaluation Environment," is described in Chapters 3 and 4. Work on Task 3, "Design of a Series of Pilot Experiments," is covered in Chapter 3. And work on Task 4, "Assessment of the Likelihood of Success," is described in Chapter 5.

The major premise for this effort is that the DoD is moving towards the large-scale use of Artificial Intelligence Systems (AISs) as tools for supporting decision-making and information analysis in military domains. It becomes increasingly important, therefore, that we develop mechanisms both for assessing the impact that these systems have upon the ability of the individual decision-makers and analysts to carry out their assignments, and for assessing the overall organizational impact that this would have. This is what we choose to call the problem of *Performance Evaluation*. The obvious question of the degree of expertise possessed by an AIS is only one small part of the evaluation problem. Without higher level, formalized evaluation techniques, complex tradeoffs of expertise level versus response time versus cost may be left up to gross estimates by DoD program managers. They must have assistance in this task, especially before commanders or the public will allow AISs to be embedded in autonomous, real-time, or strategic C^3I operations. A nearer term need is to evaluate the various AISs and AIS tools produced by DARPA programs, and in particular by the Strategic Computing program.

In Chapter 2 we described our methodological approach to the evaluation of AISs. We focus primarily on the issue of what constitutes an appropriate set of evaluation criteria, and describe a technique for organizing and manipulating such information. We are not concerned, except in passing, with actual measurement procedures (and their associated metrics) nor with mathematical techniques for computing performance evaluations from the measurements. Our view is that the proper structuring of performance evaluation knowledge is the critical step in performing an evaluation. The technique we develop is based on organizing the knowledge into *Evaluation Frames*. These are used to define the context in which the evaluation is to be performed and reflect such factors as the kind of system being evaluated, the goals of the evaluation, and the role of the evaluator. Implicit in this approach is the idea that evaluation is multi-faceted; thus different perspectives will produce different evaluations. Our techniques can easily be generalized to other kinds of information system and to other evaluation problems.

In Chapter 3 we develop a number of example evaluations that illustrate the main features of our methodology. We explore the evaluation of a number of expert system building tools (including KEE, ART, and Knowledge Craft). We perform an evaluation of the ASTA system for science and technology analysis of radar signals, and we perform an evaluation of the RUBRIC system for full-text information retrieval. The latter two systems are operational prototype decision aiding system developed at ADS. The evaluations are not intended as exhaustive analyses of the AISs with which they are concerned, but are designed to show how the concept of an evaluation frame can be used to structure evaluations. In addition, these examples help define the needs and requirements for the evaluation environment, and they serve as simplified pilot experiments for the methodology.

In Chapter 4 we describe the design of a prototype system that can support both the evaluation of AISs, and research and development into evaluation methods. We discuss a "workbench" approach to the design of the environment, describe the top-level system specification, and describe the main tools supported by the environment.

In Chapter 5 we consider the question of the likelihood of success in implementing and using the proposed environment. We conclude that our approach has relatively low technological and implementation risks, and that it can produce significant improvements in our understanding of the issues involved with evaluating the performance of AISs.

We summarize the work in Chapter 6 and make recommendations for the Phase II effort.

Chapter 2

AN EVALUATION METHODOLOGY

As AISs become increasingly accepted as tools for supporting decision-making and information analysis within military organizations, it becomes increasingly important that we develop techniques for assessing the impact that they have. Whilst it may seem natural to ask the question, Is the AIS intelligent?, we believe that this is often an inappropriate question (even if we could define intelligence), and fails to recognize the real role that AISs play in human organizations. Much more interesting, and useful, are questions such as, Does the AIS help the user?, In what ways does it help?, and, How can we quantify these benefits? That is not to say that we might not have questions about intelligence, but rather, we believe, that these will in general be a secondary issue.

To illustrate, we might be interested in high-level performance questions such as the quality of a commander's decision or the completeness of a hypothesis describing the order of battle in a complex scenario. Or we might be interested in more general questions such as the ease of use of the system independently of the range of problems it can tackle. Similarly, we might be concerned with the quality of the explanations given and the amount of time needed to learn to use the system. That an intelligent system might give "better" performance than a non-intelligent one is certainly a plausible hypothesis in these situations; however, we can see that many other factors impact the overall assessment of performance.

Our goal in this chapter is to develop a methodology for the evaluation of AISs. We will focus primarily on the issue of what constitutes an appropriate set of evaluation criteria, and describe a technique for organizing and manipulating such information. We will not, except in passing, be much concerned with actual measurement procedures (and their associated metrics), nor with mathematical techniques for computing performance evaluations from the measurements. Our view is that the proper structuring of performance evaluation knowledge is the critical step in performing an evaluation.

The first part of the chapter is concerned with nature of the evaluation process itself. The second part contains our approach to the organization of performance knowledge. The third part contains an illustrative example. And the fourth and final part discusses some needs and requirements generated by the basic concepts that we have introduced.

2.1. THE NATURE OF EVALUATION

In this section we develop our notion of an evaluation context, discuss the features of AISs that mark them as distinct from other computer-based systems, and briefly review the system evaluation literature.

2.1.1. Evaluation Context

Our approach to the problem of AI system evaluation is founded on a belief that evaluation does not take place in isolation, but is performed in order to answer a question or inform a decision. That being the case, evaluation is merely one activity, performed in conjunction with others, that ultimately leads to a decision action. The evaluation to be performed must thus depend upon the context in which it is located. The context can have many dimensions and in this section we explore four: the type of evaluation question being asked, the characteristics of the person asking the question, the kind of system being evaluated, and the stage of development of the system.

2.1.1.1. Evaluation Questions

Broadly speaking, we can identify two basic evaluation needs. In the first case, the need is to compare systems. This might be a comparison of two systems for possible purchase, or the comparison of a new system against an existing system. In the second case, the need is for an absolute assessment of performance. This is the case when a system needs to evaluated with respect to a specification or benchmark.

In practice these needs get expressed in a number of ways. In particular, they appear as questions about the performance of the AIS. Some example questions might be:

- Which system is best?
- Does this system meet my needs?
- Is this system better than the one I already have?
- Does the system do what it is supposed to do?
- What can this system do?
- What are the strengths and weaknesses of this system?

We can, of course, develop variants and extensions of these questions, but the important point is that the evaluation ultimately relates back to the question being asked. This question, in turn, characterizes the context of the evaluation.

2.1.1.2. System Evaluators

It is important to recognize that there can be many kinds of evaluators, for any given system evaluation and any given evaluation question. Each evaluator will bring a different perspective to the evaluation problem, which may translate into different sets of evaluation criteria. This implies, of course, that a system which is "superior" for one evaluator may be "inadequate" for another, simply because they emphasize different aspects of performance.

We can easily construct a list of possible evaluators. Not all situations will involve all types of evaluator, of course, but by doing this we can begin to see the range of perspectives that might be brought to bear in any given situation.

To start with, we might consider the *system developer* as an evaluator. This person is likely to emphasize criteria such as the quality and utility of tools, the elegance of the implementation of system functions, and the overall flexibility of the system code.

At a similar level, we might think of the *system maintainers* or *user support staff* as evaluators. They are likely to emphasize criteria such as quality of documentation, reliability of system code, quality of system upgrade material supplied by the original vendor, and ease of integration of the AIS software with other parts of the organizational software environment.

Next we should consider the actual *system user.* That is, the person who interacts directly with the AIS and for whom the system was (presumably) principally designed. The criteria applied here will mostly focus on the system's ability to support the user's specific task-oriented needs, and will thus be problem specific. In addition though, the user is likely to have general evaluation criteria such as the quality of the user documentation, the availability and quality of training material, and the integration of the AIS with other currently available tools.

Another interesting group of evaluators are the *managers* of the developers, maintainers, support staff and users. Overall, their evaluation criteria focus on whether the AIS enhances the productivity of the group, and whether it can respond more effectively to requests for its services.

Yet another class of management level evaluators are those that have financial responsibility, the *funders.* Of course they may also be the managers, but by adding the monetary dimension to the job description we also imply that the evaluation criteria will include some that emphasize the cost-benefit trade-offs and some that address general budgetary checks and balances.

Finally, we should consider a level of evaluation performed by a *policy maker.* Such people will typically have wide-ranging, long-term concerns that emphasize broad political and strategic issues rather than specific technical ones. Generally, these evaluation criteria will be highly subjective, but nonetheless are often of paramount importance in developing the final evaluation of the AIS.

2.1.1.3. Systems to be Evaluated

It is important to distinguish between the kinds of AIS we might want to evaluate, since different systems will have broadly different evaluation criteria. For our purposes, we will make a distinction between tools, intelligent monitoring systems, autonomous systems and decision aids. And within decision aids we will make a a further distinction between expert systems and intelligent assistant systems.

The tools category includes not only expert system shells and development environments (e.g., KEE and ABE), but also general purpose AI languages (e.g., LISP and PROLOG) and even special purpose hardware (e.g., the Butterfly machine and the Connection machine). So a tool is any software or hardware that is used in the development and implementation of an AIS. Performance evaluation of these systems will necessarily focus on technical issues such as the power and flexibility of the knowledge representation, the quality of the development environment, and the speed and efficiency of the compiled code.

The second category, intelligent monitoring systems, is for those AISs that are used for detecting unusual or emergency conditions. We think here of process monitoring systems (such as those developed using PICON), and seismic activity monitors such as that being developed at ADS in the NUCINT project (Fung *et al.*, 1987). Performance evaluation for these systems will presumably involve the system's ability to correctly classify interesting events whilst at the same time rejecting spurious non-interesting ones.

The autonomous systems category is intended for AISs that perform "independently" in some sense. That is, such systems will be totally responsible for some class of decisions, a good example being the Autonomous Land Vehicle. More generally, any system that collects information and then on the basis of this information and its own internal knowledge makes plans or decisions and then implements them, will be considered an autonomous system.

In a certain sense, evaluation of autonomous systems is easier that evaluation of tools. Because the autonomous system is supposed to perform a specific task, or set of tasks, we can envisage evaluation criteria that focus on the correct performance of that task. So for the ALV, we could ask that it successfully navigate from point A to point B across certain terrain in a specific time. Similarly, for a robot in an intelligent manufacturing facility we could specify performance in terms of the rate at which it could correctly assemble a complex piece of equipment.

Our final category, decision aids, is the most diverse. The key factor that distinguishes such systems is that they are a part of a larger environment that includes human decision-makers and information analysts. So in these cases, some person is responsible for the final decision and the AIS is simply a tool (and probably one of many tools) for assisting in the decision process. Since this is such a large class, we have chosen to divide it into two subcategories. In the first we place the classical expert consultation systems such as MYCIN and PROSPECTOR. The primary characteristic of these decision aids is that they contain significant amounts of domain specific knowledge that can be accessed by the user either to confirm existing hypotheses or generate new ones. In the second subcategory we place what we call "intelligent assistant systems." Such systems are intended to provide a broader range of support (whilst perhaps at the same time being less specialized) to the decision-maker. These assistant systems may make use of conventional expert system ideas but are intended to be highly flexible and adaptable to individual user needs.

Decision aids are, perhaps, the most difficult type of AIS to evaluate. A major reason for this is that we can no longer simply isolate the system (either physically or conceptually), but are forced to consider it in the wider context of the overall human decision-making and analysis process. This will necessarily involve many other tasks, which the decision-aid is not designed to address, that together define the overall decision-making problem. So our evaluation criteria must now include highly subjective measures such as improvement in overall decision-making ability, as well as factors such as ease of use, adaptability to the user's idiosyncratic behavior, and quality of the information displayed.

2.1.1.4. Stage of Development

Another aspect of the evaluation context relates to the stage of development of the AIS. That is, if we perform the evaluation at different points in the system life-cycle then different criteria may apply. Evaluation at the time of design will be different from that at the time of full development. Similarly, a system being developed as a research prototype will be evaluated differently from one being developed as a production system.

Notice too that the purpose of evaluation changes with stage of development. In the early work with research systems, evaluation is used to resolve design issues and to help define needs and requirements. In full-scale system development, however, evaluation is primarily used to determine progress towards the design goal, and, ultimately, to decide whether to accept the system. It may still serve to resolve design issues though. For example, alternative ways of implementing a system function may be assessed on the basis of their impact on specific system attributes.

2.1.2. Specific AI Issues

From our discussion above, it will be clear that, in a general sense, conducting a performance evaluation of an AIS is no different from conducting one for any computer-based system designed to assist with information analysis and decision making. The fact that the underlying technology is artificial intelligence rather that decision analysis, say, is perhaps a second order effect in the face of a general question about how to assess the utility of a specific decision aid. Of course, an AI approach to the problem may well be better than a traditional one, but the essential nature of the evaluation process is unchanged. That is, although there may be significant lower-level differences in the way we structure the evaluation, the main evaluation criteria and the overall methodology will remain the same.

Despite these comments, it is clearly the case that AISs have some significant characteristics that mark them as distinct from previous computer-based systems. Since our effort on this project has not focussed on specific measurement procedures or formal evaluation methods, we can only address these features at the general level. Future research will be needed to make more detailed statements. Broadly speaking then, what characterizes an AIS is that it contains knowledge and procedures for using that knowledge to solve problems, and thus from an evaluation perspective, we are concerned with evaluation criteria that could be applied to these unique components.

To illustrate, we can list a number of questions we might ask of knowledge base:

(1) *Is it built using an expressive representation?* That is, we are concerned with the ability of the knowledge representation to adequately capture the knowledge needed for the domain. We could be concerned both with power of the representation and its ease of use by the knowledge engineer or expert.

(2) *Is it complete and consistent?* In the informal sense of these terms, we would certainly like the knowledge base to have enough knowledge to solve the problems in which we are interested, and we would also like the knowledge to be

self consistent. We might also be interested in the formal logical definitions of these criteria as the apply to whatever representation is being used.

(3) *How easy is it to maintain?* Of critical importance as AISs mature, is the ease with which the knowledge base can be upgraded and modified. Although the existence of knowledge management tools will greatly assist in this task, the inherent properties of the representation ultimately dominate. Thus large rule-based knowledge-bases are potentially much harder to maintain that those that use a frame-based representation.

(4) *Is it extensible?* A related question is how easy is it to extend the knowledge to cover problems for which it was not originally designed. Again, the answer is ultimately tied to the ability of the representation to capture new knowledge, or new insights, in the domain of the AIS.

Similarly, we might ask specific questions of the inference procedures.

(1) *Do they generate sound inferences?* That is, do the inference procedures generate conclusions that are valid given the knowledge in the knowledge base. This is essentially a formal question, but one which in practice has to be treated informally.

(2) *Are the inferences intuitive.* Are the ways in which the inferences are drawn consistent with the ways in which the experts perform reasoning. The primary concern here is that the system be able to explain, in a meaningful way, the methods it used to solve problems.

(3) *Do they support a variety of reasoning paradigms?* We might be concerned with the range of reasoning methods that the AIS could support. For example, evidential reasoning, non-monotonic reasoning, reasoning by analogy, and hypothetical reasoning.

Although, we can easily list these evaluation criteria, most of them are extremely difficult to measure. Not only that, we also lack good models of how performance along these dimensions effects the higher-level performance questions we would likely ask of the AIS.

Another interesting feature of at least some AISs is that a part of the intelligence may be at the level of the user interface. That is, the system has the ability to tailor its behavior to the needs of the user. In such cases, we would certainly expect that with respect to attributes such as *user friendliness*, *explanation capability* and *adaptability*, the AIS would have the potential to significantly outperform traditional approaches. But we notice that, in terms of the evaluation methodology, the basic attributes of performance do not change. What changes are the measurement procedures we need to employ to assess the value of the attributes given that the interface is in some way intelligent.

2.1.3. Existing Approaches to Evaluation

Several techniques exist for performing the multi-attribute structuring and evaluation exercise. Perhaps the most widely known is Multi-Attribute Utility Theory (MAUT) as developed by Keeney and Raiffa (1976). This is a mathematical theory based on axioms of choice and the concepts of value functions. In principle, the most general form of MAUT can be applied to a large class of evaluation problems, but in practice several strong assumptions are usually made in order that an appropriate value function can be constructed. For example, it is usually assumed that preferences do not change as the level of fulfillment of attributes change (i.e., the preferential independence assumption). As a result, it is often further assumed that an additive utility model can be used. Many criticisms of these assumptions can be, and have been, made. However, proponents of the technique point to many successful applications where the simplifying assumptions are seen to be valid.

Attempts to resolve some of the questions raised by objections to MAUT have resulted in a variety of lesser known techniques for multi-attribute assessment. One such is the Analytic Hierarchy Process (AHP) developed by Saaty (1980). This can be viewed as an attempt to construct a hierarchical model of the decision maker's value function. It uses a two stage process to do this: structuring of attributes followed by pairwise subjective assessment of relative preference over these attributes. As with MAUT, AHP has been applied to a variety of practical problems. Objections are usually based on questions about the cognitive validity of the preference scale employed and the suitability of the linear model implied by the mathematics of the process. However, we have had some success in applying this technique to the fusion system evaluation problem (see Abram *et al.*, 1983).

The main difficulty with techniques such as MAUT and AHP as used to evaluate complex systems is that they do not employ experimental designs that provide the constraints needed to test hypotheses about the subjective scale values and combining functions used. Thus the conclusions reached by these methods rest on untested assumptions. In an attempt to overcome this difficulty Veit and Callero (1981) have developed an evaluation technique called the Subjective Transfer Function (STF) approach.

In the STF approach, transfer functions and the proposed causal structure are simultaneously verified. To do this, questionnaires are generated from factorial combinations of the input factor levels and given to informed respondents. Each item on the questionnaire is comprised of a different combination of factor levels and thus represents a different set of system capabilities. To each item the respondent judges what the outcome along the specified judgement dimension would be in that situation. When statistical analyses of the respondents' data indicate that one or more of the proposed input factors does not affect their judged outcomes, these factors are eliminated from the structure. If respondents find the task difficult because important information is missing or if much of their data is internally inconsistent, other factors are sought (usually through interaction with the respondents) to describe the structure. The design that generated the questionnaire makes it possible to test

different algebraic formulations for the transfer functions. Only after appropriate transfer functions are found to explain all the judgement data does the final causal structure emerge. Once an appropriate transfer function is found, it is possible to use it to predict what the outcome would be for each possible combination of the input factor levels and hence each different set of system capabilities.

In the recent AI literature there have been a number of discussions of the problem of evaluating and then selecting expert system building tools (or techniques). An early example is the RAND report by Waterman and Hayes-Roth (1982) which compares the performance of a number of high-level programming systems and languages. More recently, of particular interest are the papers of McCune (1984), Cromarty (1985), and Forman and Nagy (1985), which consider what features are most important in an expert system building tool. These and other papers (for example, the study of AI languages performed for DARPA by Gabriel 1985) can provide some insights into possible evaluation attributes from the perspective of an AIS builder, but do not directly address the problem of evaluating an AIS as a tool for decision-making or information analysis. The only exception we have found is the work of Gaschnig et al. (1983) which contains a good discussion of some broader issues in ES evaluation.

2.1.4. Heuristic and Linguistic Methods

The preceding comments notwithstanding, we believe that in many cases conventional methods are inadequate for expressing performance and evaluation knowledge. So in this section we present here an alternative approach which draws upon some of our earlier work in the area of linguistically based models of preference and evaluation (see Tong and Bonissone, 1980, and Efstathiou and Tong, 1982).

To help the explanation of our ideas consider the simple hierarchy shown in Figure 2-1. Here we have just two levels: an overall evaluation measure and two factors that impact the evaluation. The essence of our procedure is to ask the user to estimate the system's performance given some value of the lower-level factors. Unlike conventional methods, however, we do not use an underlying formal model to constrain the responses, but allow the user to define his or her own levels of factors and performance. Furthermore, we do not treat the responses as defining points on a function from factors to performance. Rather, in the spirit of expert systems, each estimate is represented as a rule of the form:

$$IF\ v(x_1)\ AND\ v(x_2)\ THEN\ v(y)$$

where $v(x_1)$, $v(x_2)$ and $v(y)$ denote the values of x_1, x_2 and y respectively.

The advantage of using a rule-based approach is that we can capture the subjective nature of performance in a much more direct way. In particular, we do not assume that the user's model is restricted to a limited class of linear functions. By asking for several subjective estimates of performance in several situations, and by translating this information into rules we can construct a knowledge base that can be used to make inferences about performance.

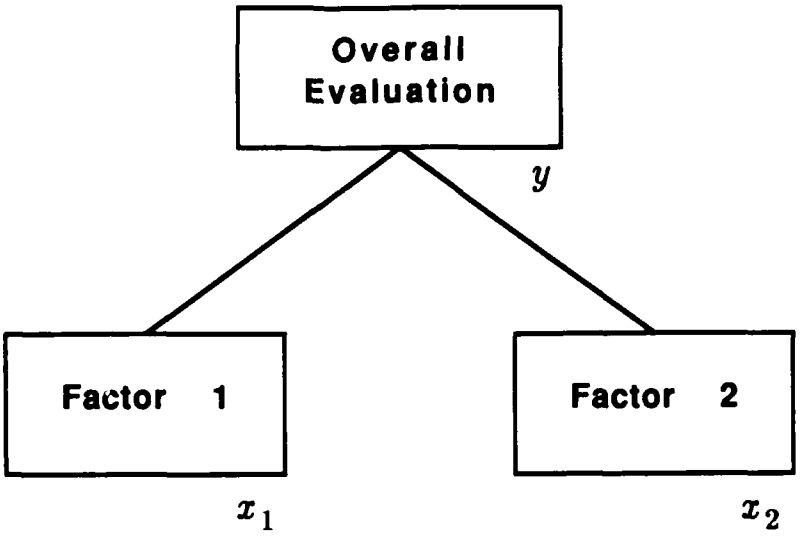


Figure 2-1 Simple Performance Hierarchy

We can generalize the applicability of the rule-based approach by allowing x_1, x_2 and y to be *linguistic variables* (Zadeh, 1975). That is, our variables can take values which are linguistic terms rather than numbers, and we interpret these terms as labels for fuzzy sets. Thus our rules become something like:

IF x_1 *is high AND* x_2 *is low THEN* y *is average*

where depending on the interpretation of x_1, x_2 and y the fuzzy terms "high," "low" and "average" will be interpreted as membership functions on some underlying spaces.

The main point here is that by making use of the principles of fuzzy set theory (Zadeh, 1965) we do not introduce arbitrary numerical precision into the user's responses, and more importantly we allow approximate matching of the antecedents of

the rules thereby providing a way of extending their scope. The results of this are that the user's knowledge is encoded in a form that is more understandable and is used in a way that parallels the heuristic reasoning processes that he or she applies. The combination of fuzzy set theory with the rule-based paradigm enables us to focus on the semantics rather than the syntactical aspects of expert knowledge and reasoning.

If we denote by $\mu_{X_1}(x_1)$ and $\mu_{X_2}(x_2)$ the fuzzy set interpretations of the linguistic terms for x_1 and x_2, and by $\mu_Y(y)$ the linguistic term for y, then formally each rule is defined by:

$$\mu_r(x_1,x_2,y) = \min\left(\mu_{X_1}(x_1),\ \mu_{X_2}(x_2),\ \mu_Y(y)\right)$$

The use of the minimum operator is not in fact the only choice, but has an intuitive appeal in that it represents a form of "independent" conjunction of the elements of the rule. Notice too that this model allows us to incorporate non fuzzy variables if we have them, since the fuzzy set membership function μ is simply a generalization of the classical notion of a characteristic function. Moreover, the variables can be continuous or discrete. (See Dubois and Prade, 1980, for a thorough introduction to the concepts used in fuzzy set theory.)

Since in the general case the user provides many knowledge fragments, and since each fragment corresponds to a rule, we need to take the conjunction of these rules to form a mapping from factors to the overall evaluation measure. So if there are N responses each of which generate a μ_r, then:

$$\mu_R(x_1,x_2,y) = \max_i \left(\mu_{r_i}(x_1,x_2,y)\right) \quad : i=1,\ldots,N$$

Notice that the use of the maximum operator amounts to saying that the output will depend on the rule that contributes most for the particular combination of variable values. (Again, this is not the only choice of aggregation operator.) If now we are given a set of factor values for some system, denoted by $\mu_A(x_1)$ and $\mu_B(x_2)$ for the pedagogical example we are considering, then by using the *compositional rule of inference* (Zadeh, 1973), we can compute an overall evaluation for the system, $\mu_C(y)$, by:

$$\mu_C(y) = \max_{x_1,x_2} \left(\min\left(\mu_A(x_1),\ \mu_B(x_2),\ \mu_R(x_1,x_2,y)\right)\right)$$

An important point here is that μ_A and μ_B do not have to be the same as the antecedents in any of the rules; indeed it is exactly this ability to provide partial matching of antecedents that gives the fuzzy set approach its power. In general when constructing rules to express evaluation structures it is advisable to give the expert a finite set of primitive linguistic values, together with some appropriate connectives, so that a rich enough vocabulary of values is available. We denote the vocabulary for variables x_1 and x_2 by V_1 and V_2, and the vocabulary for y by V_Y.

Since the computed membership function for the overall evaluation, μ_C, is not necessarily a normal function (i.e., $\max_y \mu_C(y) < 1.0$) then we can normalize this by

forming $\mu^*_C(y)$ defined by:

$$\mu^*_C(y) = \frac{\mu_C(y)}{\max_y \mu_C(y)}$$

In general, $\mu^*_C(y)$ will not correspond to one of the elements in V_Y, but by using a technique called *linguistic approximation* (see Bonissone, 1978) we can, if necessary, generate a linguistic label for $\mu^*_C(y)$ that characterizes it in terms of the elements in V_Y.

We can now see that this procedure for reasoning with the performance and evaluation knowledge is actually a very general form of uncertain reasoning that goes beyond the ideas used in more conventional rule-based expert systems. By allowing more general types of variables we considerably extend the scope and flexibility of rule-based reasoning, while at the same time providing more satisfactory models of preference knowledge.

Another basic feature of this approach is that the rule of inference can return a value of "unknown." This happens when the low-level factor values are considerably different from any of the antecedents in the response set. The advantage of this is that the modeller then gets a chance to explore his responses and either change them so that they can cope with the new situation or add additional rules. We believe that this interactive feature considerably improves our ability to build good models of performance.

Two other features of this approach are worth noting. First, we do not assume an arbitrary class of models that the user's knowledge is supposed to obey. Indeed, we view the task not as one of mathematical modeling but as one of knowledge representation. By providing a more expressive language than that provided by formal methods we are able to capture this knowledge more accurately. We lose the analytic power of the mathematical methods by doing this, but we gain in comprehension.

Second, by admitting the concept of a linguistic variable and its interpretation through fuzzy set theory, we provide a way of making explicit the inherent vagueness of subjective models. Of course, by doing this we produce vague evaluations, but we believe this is exactly what is required. At the levels of performance evaluation this techniques is designed to address, it is generally infeasible to ask for precise measures.

2.2. ATTRIBUTES, FRAMES AND PERSPECTIVES

In our view, performance evaluation consists of three basic activities; structuring, measurement, and interpretation. So the first step must always be to develop a structured set of performance criteria that reflect the important elements in the evaluation problem. Next, actual measurements of the AIS must be taken. These can be measurements in the traditional sense of speed or storage space, but can also be much softer, subjective measurements such as the "feel" of the user interface or the flexibility of the tool. Measurement may also require much more elaborate procedures to generate the necessary information; we think here of simulations, benchmark

testing, and controlled experiments. Finally, the measurements must be interpreted in the context of the structure to produce an evaluation. This might be a single number, as in traditional analyses, but more usefully might be a vector of numbers that represent various aggregates of performance criteria.

As we have indicated in the introduction to this chapter, we consider the structuring issue to be paramount. Consequently, in this section we develop a technique to support the representation and organization of performance knowledge. Our goal is to define an integrative framework within which this performance knowledge can be used together with measurement and interpretation procedures to generate meaningful evaluations of AISs.

2.2.1. Evaluation Attributes

The atomic concept on which our approach to evaluation is built is that of the attribute. An attribute is any characteristic of the AIS that is potentially relevant to the assessment of performance. For example, we can identify some rather generic characteristics such as content, form and process that allow us to begin the development of a more detailed evaluation model. Attributes of content relate to the problem solving components of the AIS, and might include the completeness of the knowledge contained in the AIS and the ability of the knowledge to be generalized to other problems in the same domain. Attributes of form are concerned with the interaction between the AIS and its environment. They include characterizations of the user-interface, but might also be concerned with the expressive power of the knowledge representation language. Attributes of process are primarily system oriented. Such attributes might include the speed of response to requests for information, or the capability of the AIS to explain, or expand upon, conclusions when so requested.

In our model of evaluation, attributes are to be thought of as the basic building blocks from which to construct an evaluation structure. That is, they form a space of primitive elements; the *Attribute Space*. To be used in an evaluation, the attributes need additional information to be attached to them (e.g., a measurement procedure, preferences) that will, in general, reflect the context of the particular evaluation. So, attributes by themselves can be thought of as context-free entities which when given a context generate specific instantiations of themselves.

As an example, let us consider the problem of an AIS for full-text document retrieval (see the following chapter for a detailed evaluation of such a system). An attribute of performance that applies to all document retrieval systems is "precision". This is a measure of the accuracy of the retrievals produced by the system, and has a formal definition in terms of the ratio of the cardinality of two sets. Notice that this information is independent of any particular system, or document collection, or information retrieval technique. Precision is a context-free attribute. Another such attribute for document retrieval systems is "recall". This is a measure of the comprehensiveness of the retrievals produced by the system, and also has a formal definition. Together, these two attributes define another attribute, "retrieval

effectiveness'', that is also context-free. What distinguishes this compound attribute from its components is exactly that it is defined in terms of them. Notice, however, that the form of this dependency is left unspecified; such information will result from defining a context.

The attribute space is thus a partially ordered set of entities, where the ordering relation has the semantics of ''is a function of''. In building an evaluation structure, naming a compound attribute also names its components.

2.2.2. Evaluation Frames

Our discussion in the first part of the chapter described some of the dimensions of the evaluation context; the evaluation question being asked, the characteristics of the evaluator, the kind of AIS to be evaluated, and the stage of development of the AIS. To help separate out these effects, we introduce the notion of an *Evaluation Frame.* We think of this frame as defining the context in which the evaluation of the AIS can be performed. As we have indicated, there are many possible ways in which we could specify the contextual parameters of the frame. The dimensions we have outlined, or some cross-product of them, provide a useful initial set of frames. However, the intent is that the frame defines any useful evaluation context. Irrespective of the actual definitions, the purpose of the frame is to allow us to focus our evaluation by collecting together significant pieces of evaluation knowledge. That is, once the context has been specified we can begin to ask more detailed questions about the evaluation criteria that should be applied.

Evaluation frames consist of five basic elements: a list of attributes, an evaluation structure, preferences, measurement procedures, and interpretation methods. Let us consider them in turn.

Attributes are, of course, the instantiated versions of the elements in the attribute space. The context will define which attributes are broadly relevant and so to be included in the list. That is, the type of system being evaluated and the characteristics of the evaluator will generally dominate the choice of attributes. User frames will not usually contain attributes associated with the cost of the system, manager frames usually will. Similarly, frames for decision aids will certainly contain user interface attributes, frames for autonomous system will probably not.

The evaluation structure corresponds to context-dependent connections between attributes. So in our document retrieval example, we might have specified that *retrieval effectiveness* and *retrieval efficiency* (i.e., the speed of response of the system to a user query) are to be included in the frame, and that together these define a new context-dependent evaluation criteria called *retrieval performance.* That is, *retrieval performance* is a function of *retrieval efficiency* and *retrieval effectiveness.* It is this additional dependency information supplied as part of the specific evaluation context that we call evaluation structure.

Preference information supplied in the frame is concerned with specifying the relative importance of the attributes within the overall evaluation. So in our running

example, it would correspond to defining the trade-offs between efficiency and effectiveness, and then between precision and recall, for determining the overall retrieval performance.

The process of measurement involves subjecting the AIS to a variety of tests and reporting the results. In most cases, the actual procedures to be used are specified as part of the frame and are attached to the appropriate attributes. In our example, we would need to specify the document collections and example queries, and methods for reporting the results. In general though, as we noted above, such tests may involve measurement of engineering parameters, or subjective evaluations of higher level system characteristics. For some basic attributes of all systems, such as speed of response, we note that the measurement procedure may actually be thought of as being attached to the attribute itself.

Notice that in this study whilst we recognize that testing may require complex procedures to be performed in order to get the required parameter values, our focus is not on these procedures *per se* but on their ability to generate useful measurements. Thus we are interested in the range and type of values that the measuring instruments can return, and in particular we are interested in the accuracy with which the measurements can be performed.

The final piece of the frame is the interpretation method (or possibly methods). Measurements are the data on which we will build the evaluation, but it is the interpretation method that allows us to actually manipulate them. Of course, if the evaluation knowledge is encoded algorithmically (as it would be if we were using a conventional evaluation technique) then this is simply a question of executing the algorithm. However, if we have encoded this knowledge using an AI derived representation (e.g., our extended rule-based technique), then interpretation is a question of performing forward inferences from the data to the final evaluation. We note that this will usually involve additional kinds of knowledge (i.e., control knowledge) which ensures that the basic evaluation knowledge is applied correctly. A useful analogy is to think of the complete frame as a piece of executable code.

An important question in the interpretation phase is how to combine the evaluation information generated by the sub-structures within the overall evaluation structure. There can be no general solution to this problem but there could obviously be some default procedures with special cases being identified as necessary. The knowledge necessary to support this is encoded at the frame level as part of the interpretation method itself. In the case where there is a single interpretation method for the frame, the combination operations are given as part of the definition of the method. In the case where there are several methods, the builder of the frame must specify how the evaluation information is to be combined. One possibility for resolving this issue is to allow attributes to inherit the method from the frame. The user would then simply select a method at the frame level and the appropriate information would be attached to the attributes. The advantage of this approach is that the frame itself could have several methods attached to it, with the user able to switch between them as needed.

2.2.3. Evaluation Perspectives

An additional feature of our representation is that it provides for the construction of higher-level evaluation structures that we call "perspectives". A perspective is similar to a frame in that it has structure, preferences, and interpretation methods. What makes it different is that the evaluation primitives are frames rather than attributes. Thus we consider a perspective to be a combination of evaluation frames that when taken together give a more complete picture than the individual frames alone. So for example, we might have a group of users trying to select a system for a specific group activity. Each individual in the group has an evaluation context (a personal frame) but the overall evaluation should be performed at the group level (the group perspective). Note that the perspective does not necessarily attach to a particular evaluator but merely represents some appropriate grouping of evaluation frames. Thus a perspective contains evaluation information that is primarily concerned with describing how the evaluation produced by the individual frames should be combined into an overall evaluation.

Actually, our definition of a perspective can be extended so that an element of a perspective can be a lower-level perspective as well as an individual frame. Thus we have the ability to combine evaluations from different perspectives into higher level perspectives, and so generate complex evaluation hierarchies.

2.3. AN ILLUSTRATIVE EXAMPLE

In this section we will develop a simple example that serves to illustrate the ideas described above. Suppose you are the manager of a small department that is considering the purchase of an AIS to support some technical activity. (For the purposes of this illustrative exercise we do not need to specify the actual application.) As the manager you have a number of evaluation criteria as illustrated in Figure 2-2.

This Figure shows that evaluation at the department level is to be considered as a perspective. That is, an overall evaluation of the AIS is a combination of the output from the manager's own evaluation frame (shown dotted in the Figure) and the output from perspectives generated by two classes of user in the department. These users are characterized as either "novices," in which case they are unfamiliar with the problems encountered and problem solving techniques used in the department, or "experts," in which case they are very familiar with the problems and techniques.

Before considering how the novice and expert perspectives are generated, we shall explore the manager's evaluation frame. For the purposes of the example, we assume that system utility is a function of only two attributes: cost and customer support. System utility is thus a compound attribute, and cost and customer support are primitive attributes. Let us look at these in turn:

System Utility. This is the overall characterization of the system being evaluated and represents the general suitability of the system within the context of the current evaluation frame. For this example we think of it as taking numerical values on some utility scale, the interval [0,100] say.

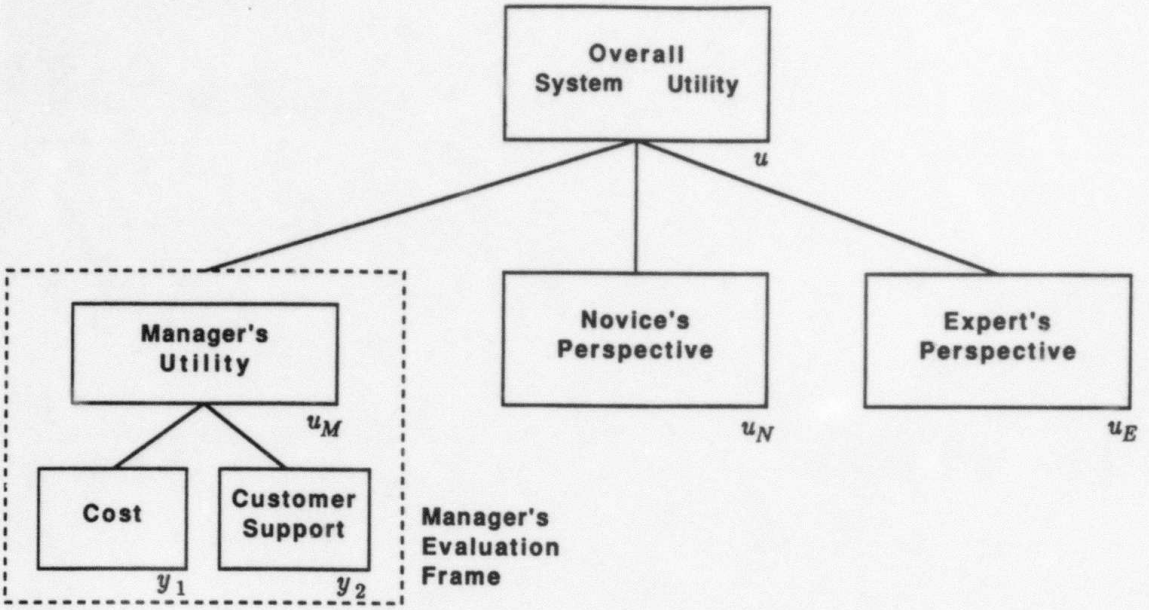


Figure 2-2 The Departmental Perspective

Cost. This is intended to be the actual dollar cost of purchasing the AIS.

Customer Support. This attribute captures the manager's concern with the quality of the support provided by the vendor of the AIS. Appropriate linguistic values might be "good," "average" and "poor."

In terms of our methodology, we can think of this step as the result of copying two attributes (cost and customer support) from the attribute space, instantiation by specifying the type and range of the values associated with these attributes, and then adding evaluation structure to generate the context-dependent attribute, system utility.

The next step is to specify preferences, measurement procedures and interpretation methods for the frame. We will assume that the manager is able to construct a simple additive utility function of the form:

$$u_M = \alpha\, y_1 + \beta\, y_2$$

so that this single equation defines both the preferences (the parameters α and β) and the interpretation.

Since this is a pedagogical example, we assume that the measurement procedures attached to the primitive attributes are straightforward to specify. For cost, the manager must ascertain the cost, presumably by looking at the sales literature or talking to the salesperson, and then report it. For customer support, the manager must generate a subjective impression based on information given by the vendors, conversations with existing customers of the product, and any other sources of information that she has, and then map this onto the variable y_2.

To complete the definition of the departmental perspective, we need to specify how to combine the value from the manager's own frame with the values from the novice's perspective and the expert's perspective. To do this we assume that this knowledge is specified heuristically as:

IF $u_M \leq 30$ or u_N is low
THEN u is low

IF $u_M \geq 75$ and either u_N is high or u_E is high
THEN u is high

Notice that in deriving the overall evaluation, the departmental perspective emphasizes the manager's and the novice's input more than the expert's. That is, the overall structure of the perspective permits the manager's and the novice's evaluations to dominate the final evaluation. Of course, these rules are extremely simplistic and in a real example many more would be needed in order to be able to compute the final perspective evaluation. We use these two rules simply to illustrate how we can combine more than one evaluation method.

Let us now consider how to determine the perspectives of the novice and expert problem solver. Both of these users make use of the same attribute space which contains the elements shown in Figure 2-3. The figure shows three attributes, namely *Ease of Interaction*, *Generality of Knowledge* and *Explanation Capability*, as examples of general features that might be considered important in a real evaluation. *Ease of Interaction* is itself a function of the primitive attributes *Graphics Utilization* and *Input Devices*. Let us consider these attributes in turn.

Ease of Interaction. This attribute describes the ease with which the user can interact with the system. It is very much a subjective notion and might be characterized in linguistic terms such as "excellent," "good" and "poor." Notice that in a real system the selection and calibration of these primary vocabulary terms is an important part of setting up the evaluation frame.

Graphics Utilization. This attribute represents the degree to which the AIS makes use of graphics, and combines the extent of the graphics support provided by the system with the extent to which the graphics are used to enhance the presentation of information. Linguistic characterization might be in terms such as "high," "medium"

1. Ease of Interaction
   1.1 Graphics Utilization
   1.2 Input Devices

2. Generality of Knowledge

3. Explanation Facilities

Figure 2-3 An Attribute Space for Problem Solvers

and "low."

Input Devices. This attribute simply describes the input devices available and in our simple example can take one of two values: "keyboard only" or "keyboard and mouse."

Generality of Knowledge. This attribute is concerned with the extent to which the knowledge on the system can be used for a variety of problems. If there were a fixed set of problems of interest then the underlying scale could be a percentage and the linguistic values might be something like "covers most problems," "covers a subset of the problems" and "covers a small fraction of the problems."

Explanation Facilities. This attribute describes the degree of explanation that the system can provide in response to a user request. It includes the extent of the on-line help as well as the form of explanation given for a particular conclusion. Linguistic

characterization might be in terms such as "excellent," "good" and "poor."

To construct the novice user's evaluation frame we need to copy the appropriate attributes from the attribute space, add an evaluation structure and then define the preference information and interpretation methods. This information is summarized in Figures 2-4 and 2-5. Notice that the frame structure generated by the novice does not include the *generality of knowledge* attribute (x_2) since in the early stages of becoming familiar with the system the novice is less concerned with the problems the system can solve than with the ease of use and the help it can give. Notice too that the novice is especially concerned with the explanation facilities; if they are poor then the AIS will get a low rating.

The expert user's evaluation frame is summarized in Figures 2-6 and 2-7. Notice that the expert is very concerned with the range of problems that the system can handle, and if this is only a small subset of those of interest then the system would not be perceived as very useful (see rule 2 in Figure 2-7). Also, the expert, unlike the novice, is somewhat less concerned with ease of interaction (variable x_1) and is willing to accept a lower level of graphics utilization and simpler input devices provided that the system can handle the problems of interest.

To create perspectives we would take the evaluations from several novice and expert users and combine them using an appropriate interpretation method, thus generating an overall evaluation for each of the two user groups. We omit that step here, and proceed to the final overall interpretation. We will assume that the AIS had the following characteristics:

x_{11} is *medium to high*
x_{12} is *keyboard only*
x_2 is *approximately 20%*
x_3 is *excellent*

Then using the fuzzy set operations described in the Section 2.1.4, and assuming appropriate characterizations of the linguistic terms, we can infer that the novice group and the expert group will have evaluations that rate our imaginary AIS as *between medium and high* in the case of the novice, and as *more or less low* for the expert. If we look at the knowledge bases for these two types of users we can immediately see the reason for this; the novice needs good interactive capability and good explanation facilities but the system is only capable of a *medium* value for x_1 which in turn impacts the overall rating, and the expert above all needs good coverage of the problem domain but the system has only *approximately 20%* coverage which is only marginally offset by the *excellent* value for x_3.

We can now complete the evaluation by feeding these values into the departmental perspective. If we assume that cost (variable y_1 in Figure 2-4) is \$15,000 (a moderate value) and customer support (y_2) is *good* then using the utility equation given earlier (with appropriate values for the preference parameters α and β) we get a system evaluation for the manager (u_M) of *high*, which in turn leads to an overall evaluation of *slightly above medium* for our imaginary AIS. Figure 2-8 shows the fuzzy

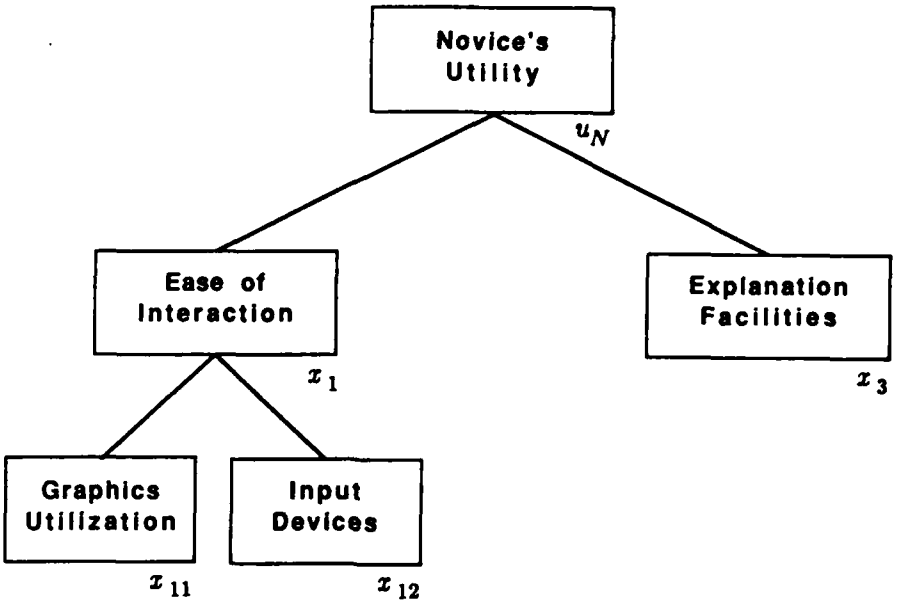


Figure 2-4 Novice's Frame Structure

1: IF x_1 is excellent & x_3 is excellent
THEN u_N is high

2: IF x_3 is poor
THEN u_N is low

3: IF x_{11} is medium & x_{12} is keyboard
THEN x_1 is medium

4: IF x_{11} is high & x_{12} is keyboard and mouse
THEN x_1 is excellent

Figure 2-5 Novice's Interpretation Knowledge

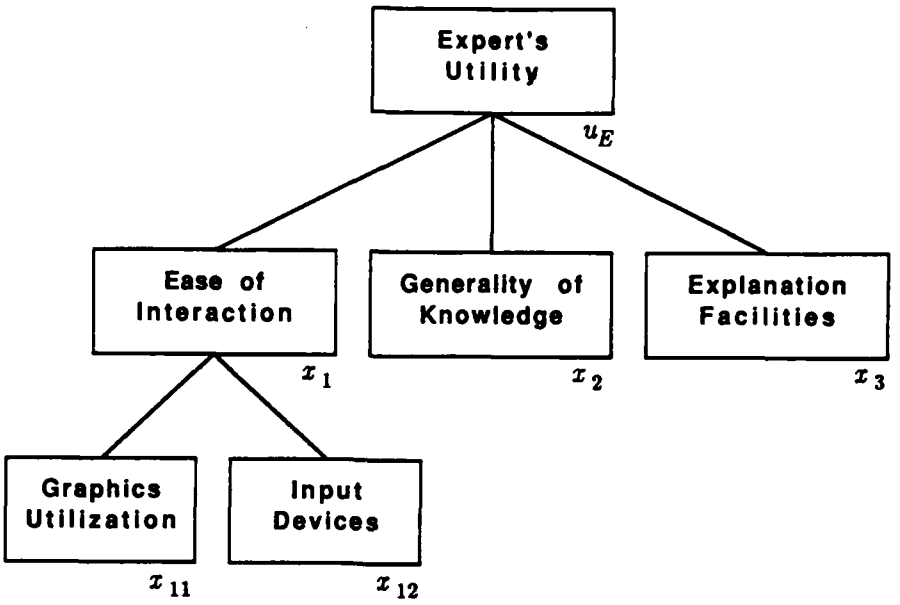


Figure 2-6 Expert's Frame Structure

1: IF x_1 is good or better & x_2 is most problems & x_3 is good or better
THEN u_E is high

2: IF x_2 is small subset only
THEN u_E is low

3: IF x_1 is poor
THEN u_E is low

4: IF x_{11} is high & x_{12} is keyboard
THEN x_1 is excellent

5: IF x_{11} is low & x_{12} is keyboard and mouse
THEN x_1 is poor to good

Figure 2-7 Expert's Interpretation Knowledge

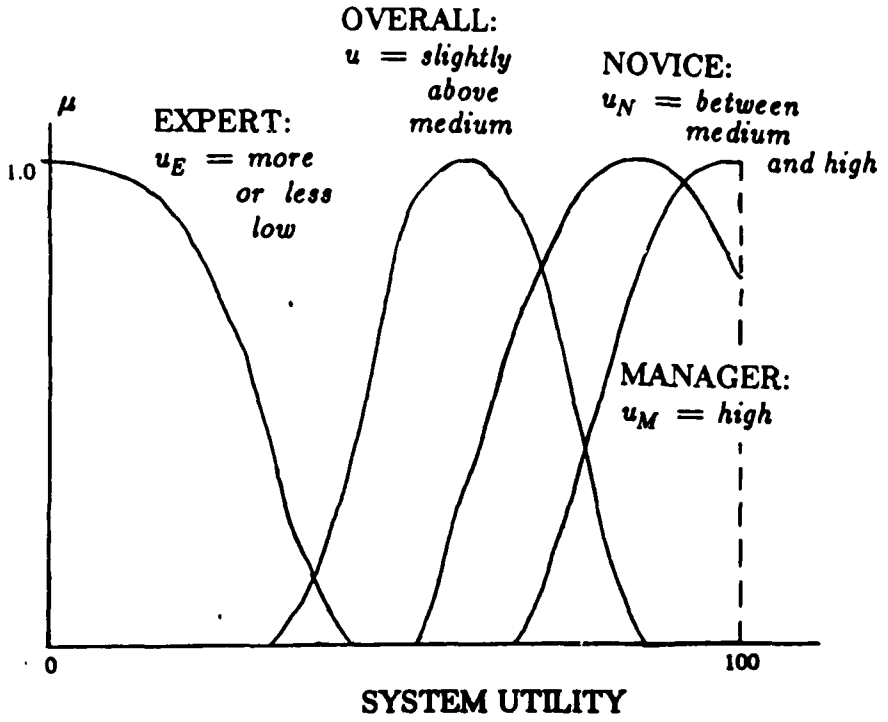


Figure 2-8 Example Evaluations for the Novice, Expert and Manager

set interpretation of the linguistic ratings generated by the novice's perspective, the expert's perspective and the manager's frame, as well as the overall evaluation derived from the departmental perspective. Notice that, as we would expect from the definition of the departmental perspective combination operation given earlier, the expert's evaluation is partially discounted generating an overall evaluation that is closer to the novice's.

2.4. EXTENSIONS TO THE BASIC METHODOLOGY

Clearly the example just presented is much simplified and is not intended as a demonstration of the full power of our approach. It is intended to show the high-level operation of our methodology, the nature of the information needed by such a methodology and the benefits to be gained from a structured approach to modelling preference and evaluation knowledge. In practice, we would expect that performing an

evaluation of an AIS using our methodology would be an iterative process involving detailed analysis of intermediate evaluations and revisions of the evaluation frames. To support this interaction with the evaluation process, the basic methodology requires some extensions and additional features. We briefly mention them here.

It is clear that we need to provide a capability to create libraries of objects used in the evaluation process. We will need libraries of attributes, frames, perspectives, and interpretation methods, together with tools for managing these knowledge bases. Such tools might be broadly characterized as browsing tools. That is, they allow the evaluator to explore the contents of the libraries and select items as needed for the evaluation.

A second need is the capability to create and modify evaluation objects. A general purpose editor is required that can help the evaluator perform all of the necessary tasks such as entering new elements in the libraries, creating and tracking modifications to basic objects, and actually executing evaluations.

Finally, the results of the evaluation have to be presented to the evaluator, or to the person who is acting for the evaluator. A requirement at this point is the ability to display evaluation results in a variety of ways; simple numerical information, trade-off curves, explanation traces, sensitivity analyses, and what-if? scenarios. Thus the final evaluation is not just a case of "looking at the numbers", but is itself a process that encourages the evaluator to explore the content of the evaluation in a variety of ways. In particular, notice that although we have generated one overall rating of the example AIS, the fact that we have a structured set of frames and perspectives allows us to generate many possible trade-offs between evaluation attributes.

Chapter 3

EXAMPLE EVALUATIONS

In this chapter we develop an number of example evaluations that illustrate the main features of our methodology. These are not intended as exhaustive analyses of the AISs with which they are concerned, but are designed to show how the concept of an evaluation frame can be used to structure evaluations.

3.1. EXPERT SYSTEM BUILDING TOOLS

Our first example is concerned with an evaluation of a number of expert system building tools (ESBTs). We have used a recent article by Gevarter (1987) as the starting point for our analysis. This article contains an extensive characterization of twenty commercially available ESBTs although we make no claim, of course, that it is a consistent and complete characterization. Indeed, as we shall see in Section 3.1.4, each application may have specific performance needs not covered by such a general purpose set of attributes. In such cases, the basic attribute space needs to be augmented appropriately.

3.1.1. ESBT Characteristics

In his article entitled "The Nature and Evaluation of Commercial Expert System Building Tools", Gevarter identifies a number of ESBT characteristics that can serve as the foundation for a general attribute space. These are summarized in Figure 3-1. Notice that in general his characterizations are actually labels for *measurements* of ESBT features and are not performance attributes in the sense that we use that term. For example, a list of knowledge representations that the ESBT supports is a measurement not a statement about evaluation. Such information can be used to perform an evaluation, of course, but by itself tells us nothing about the value of the ESBT. The two evaluation frames that we construct in the following sections illustrate how this basic measurement data can be utilised to perform ESBT selection.

3.1.2. A Neophyte Expert System Builder

Our first evaluation frame is constructed around the needs of an engineer in a medium sized company who has several years experience with conventional software engineering methods, but who wishes to become familiar with tools for building expert systems. Her goal is to find an ESBT that can be used on a PC-style computer and which costs as little as possible consistent with it having a number of knowledge representations and inference methods. In addition she would like a tool that has a flexible style of interaction; for example, one that supports both mouse and keyboard command entry. Notice, that the goal is not to find a tool to build an ES for a particular application, but rather to become familiar with as wide a range of ES building techniques as is possible. Presumably, once this additional experience has been gained, she will be in a position to make a more informed decision about tools for actual projects.

1. Knowledge Representation

2. Inference Method

3. Pattern Matching Method

4. System Developer's Interface
   - 4.1 Knowledge Acquistion
   - 4.2 Knowledge Management
   - 4.3 Explanation
   - 4.4 Display Development
   - 4.5 On-line Help

5. Software Environment

6. Hardware Environment

7. Integration with External Systems

8. End User Interface

9. Cost

Figure 3-1 ESBT Characteristics

The attributes that apply to this frame are summarized in Figure 3-2. Thus the ESBT must be compatible with the hardware and price constraints imposed by the user, must provide multiple inference and knowledge representation methods, and must allow multiple interaction styles. Notice that the hardware constraint is indeed a hard constraint and as such serves to prune the space of alternatives. That is, it acts as a necessary attribute which if satisfied plays no further part in the evaluation. Information on this can be extracted directly from Gevarter's data, as can the price information. We can derive a measure for inference methods and knowledge representations simply by counting the number of methods that the ESBT supports; this also can be extracted from Gevarter's data. Information on the interaction styles is unfortunately not present in Gevarter's data, but as we shall see it is really a secondary factor and is only required to break a tie between two tools.

1. Hardware Compatibility

2. Cost Compatibility

3. Support for Multiple Inference Methods

4. Support for Multiple Knowledge Representations

5. Support for Multiple Interaction Styles

Figure 3-2 Neophyte Expert System Builder's Attributes

Applying the hardware constraint results in a list of twelve ESBTs. In Table 3-1 we list them together with their performance with respect to the cost, inference and knowledge representation attributes. Notice that under Gevarter's scheme the maximum number of inference methods is six and the maximum number of knowledge representations is four.

At this point, the decision has become relatively straightforward. Two ESBTs, ESP and 1st Class, both dominate the rest of the tools with respect to the number of inference and knowledge representation methods, and both cost less than one thousand dollars. Thus we see that in this case there was no need to construct a formal preference structure over the complete attribute set in order to get a tractable decision problem. To make the choice between the final two options, the user must simply trade-off cost (i.e., the difference between $900 and $500) against interface

flexibility. In this example, further investigation by us* revealed that ESP has multiple interaction styles whereas 1st Class has essentially just one. At this point it would be possible to perform a formal analysis of preference (e.g., construct a two variable multi-attribute utility function), but the decision might simply rest on whether the user herself or her company is paying for the ESBT. We might presume that if the former then the choice would be 1st Class, and if the latter then ESP.

3.1.3. The FRESH Tool Evaluation Revisited

Our second evaluation frame is based on a study of ESBTs performed by Texas Instruments Inc. as part of DARPA's FRESH project (Wall et al., 1985). Our goal

Table 3-1 Evaluation of ESBTs

Tool	Price	Inference Methods	K-R Methods
KES	$4,000	3	1
M1	$5,000	4	1
Nexpert	$3,000	3	2
PC+	$3,000	3	2
ExSys	$400	2	1
Expert Edge	$2,500	1	1
ESP	$900	4	3
Insight	$500	2	1
TIMM	$1,900	2	1
Rulemaster	$1,000	3	2
KDS	$1,500	4	2
1st Class	$500	4	3

* A series of telephone calls to the ESBT vendors.

here is not, of course, to re-interpret the findings of that report, but simply to show how our methodology could have been applied to that problem.

The TI evaluation team started by developing a set of problem specific requirements and then listed a set of evaluation attributes derived from them. In Figure 3-3 we have summarized the organization of these criteria. We see that they can be divided into four main groups: those that address knowledge representation, those that address reasoning and inference, those that address ease of engineering, and those that address systems support questions. The reasoning and inference group may be further sub-divided into criteria that assess the tool's ability to support hypothetical reasoning, the types of control provided, and the type of explanation capability.

1. Knowledge Representation

2. Reasoning and Inference
   - 2.1 Hypothetical Reasoning
   - 2.2 Reasoning Control
   - 2.3 Explanation Capability

3. Ease of Engineering

4. Systems Support

Figure 3-3 FRESH Tools Attributes

To construct an evaluation frame we must have measurement procedures and preference structures, and for the TI team this was achieved by first defining the problem requirements in terms of specific features of the ESBT tool. So under the *Knowledge Representation* attribute the TI team determined that the tool should:

- provide a mechanism for grouping pieces of related information into a structure with slots and values (i.e., a schema or frame).
- support an ability to structure the schemata into conceptual abstraction hierarchies with dynamic inheritance.
- have some means of noticing and acting upon changes in slot values.
- support some ability to describe how, and under what conditions, information can be inherited from one schema to another.

Similarly under the *Hypothetical Reasoning* attribute the tool should provide some way of temporarily changing the knowledge base as well as providing an ability to generate multiple alternative temporary hypothesis spaces. In particular, the tool needs to be able to change or add facts and relationships. Under *Reasoning Control* the tool must support data-directed and goal directed reasoning. The inference engine must also be well integrated with the knowledge base and must be able to access the schemata and constraints as needed.

The *Explanation Capability* is defined in terms of providing the developers with access to the proof trees used by the tool. From these trees it should be possible, in principle, to generate descriptions of the reasoning used to reach conclusions.

The *Ease of Engineering* attribute is concerned with whether the underlying paradigms in the tool are understandable as a complete whole, whether the tool is well integrated with the Lisp Machine environment, whether the tool has knowledge base construction aids and knowledge base maintenance aids (such as tree displays of hierarchies, knowledge base consistency checks), and whether the tool has tracing and debugging aids.

Finally, the *System Support* attribute requires that the tool run in CommonLisp on a Symbolics Lisp Machine, that the vendor provide the appropriate level of support for the tool, and that the vendor should provide training courses and consultation services.

Each of the terminal attributes is thus expressible in terms of some measurable tool characteristics. These are shown in Table 3-2, where the actual evaluations for the tools are also given. Notice that the TI team did not, in general, attempt to quantify their responses to the measurement questions. For example, all three tools are simply marked as having System Defined Relations even though there are variations within them. In fact a reading of Wall et al. would seem to indicate that for this attribute the tools could have been ranked in the best-first order ART, Knowledge Craft, KEE.

This hierarchy of factors represents only a taxonomic view of the evaluation knowledge and reflects neither the relative importance of the factors nor any necessary conditions that the evaluators have attached to individual attributes. In fact, in this

Table 3-2 FRESH Tools Attributes and Measurements

Attribute	KEE 2.0	ART	Knowledge Craft
Knowledge Representation			
Schemata	Yes	Yes	Yes
System Defined Relations	Yes	Yes	Yes
Demons	Yes	Yes	Yes
User Defined Relations	Not built in	Not built in	Yes
Hypothetical Reasoning			
Change Facts in Context	Built in Lisp	Yes	Yes
Change Relationships in Context	No	Yes	Yes
Reasoning Control			
Data-Directed Reasoning	Yes	Yes	Yes
Goal-Directed Reasoning	Yes	Yes	Yes
Integration with Schemata	Yes	Yes	Yes
Explanation Capability			
Access to Proof Trees	Vendor supplied	Vendor supplied	Vendor supplied
Ease of Engineering			
Integration	Good	Good	Good
Interface	Excellent	Good	Slow
Construction Aids	Yes	Use Zmacs	Use Zmacs
Knowledge Base Display	Yes	Yes	Yes
Tracing and Debugging	Yes	Yes	Yes
Systems Support			
Symbolics Common Lisp	Yes	Yes	Yes
Vendor Support	Established product	Recently released	Beta version evaluated
Training and Consultation	Yes	Yes	Yes

case the evaluators had two highly critical needs that dominated the evaluation. First, it was required by the evaluators that the tool be able to support user-defined inheritance, and second that the tool be able to support hypothetical reasoning (i.e., to be able to ask various forms of *what if?* question). On this basis the KEE tool was essentially eliminated, even though it performed very well on other criteria (e.g., it dominates on ease of engineering).

A particularly interesting aspect of this evaluation study is that the remaining choice between ART and Knowledge Craft was resolved by introducing an additional evaluation criterion. The evaluators decided that Knowledge Craft was inherently superior from the perspective of providing "a robust maintenance environment." That is, having determined that both the remaining tools were similar in capability, the final choice was resolved in favor of Knowledge Craft because it better supports "the size and complexity of the Navy domain" and its inherent semantics help minimize the problems that arise from "the fact that long term maintenance will be performed by people other than the original knowledge engineers." We can think of this as the creation of a subordinate evaluation frame (i.e., one that focusses on knowledge management and maintenance) that is conditional on satisfactory performance evaluation in the context of a primary frame.

We see then that this interpretation was rather complex; certainly not a simple forward propagation of measurements through the attribute structure to final evaluation, but rather a conditional evaluation based on reaching necessary thresholds on certain key criteria. It is interesting to note that the presence of these key criteria, if they were recognized as such at the beginning of the evaluation exercise, could have served to direct the measurement process. For example, we can see that the Ease of Engineering and Systems Support attributes were basically ignored in the final evaluation. At least for the former, the evaluators went to a lot of trouble to develop simple prototype systems using each of the tools in order to generate the assessments. However, the criteria that the tool provide user definable inheritance relations and hypothetical reasoning could probably have been easily checked by reading the tool specifications. Such a strategy might have reduced the evaluation costs substantially. Similarly, if the knowledge maintenance issue had been initially recognized as critical, then ART's reliance on a rule-based representation might have eliminated it from consideration without having to have performed any other evaluations.

An interesting question now is how we might map the TI evaluation team's attributes onto the attribute space defined by Gevarter's analysis. What we find it that there is not a one-to-one correspondence between them. In particular, Gervarter does not have an attribute that addresses the need to support user defined inheritance, nor does he clearly address whether the ESBTs can support the specific kinds of hypothetical reasoning that the FRESH problem requires. However, we can perform an approximate mapping if we make some simplifying assumptions. Specifically, we shall assume that for the purposes of this revised evaluation frame that we can ignore the system support attributes (otherwise we find that KEE, ART and Knowledge Craft are in fact the only ESBTs that run in CommonLisp on a Symbolics). We shall also

assume that Explanation Capability can be represented by a count of the features labelled by Gervarter under his Developers Interface criteria as Why, How and Explanation Expansion. Similarly, we will represent Ease of Engineering by a count of Gevarter's attributes Check for Consistency, Graphics Representation of the Knowledge Base, and Inference Tracing.

Given these assumptions we get the evaluations shown in Tables 3-3a, 3-3b, 3-3c and 3-3d. As we might expect, KEE, ART and Knowledge Craft appear to dominate this list, and of these, KEE appears to be the "best" since it scores highest on the Ease of Engineering count. However, since, as we pointed out in the preceding discussion, there is not a direct correspondence between attributes in these two frames, all we can reasonably conclude is that one of these three tools should be selected. Interestingly, though, the evaluations using this frame suggest that one of the less sophisticated tools, namely Nexpert, might have capabilities that approach those needed for the FRESH problem. All it appears to lack is the capability to reason with contexts! Furthermore, if we completely relax the requirement to do hypothetical reasoning then we find that PC+ and ESP are also potential candidates.

Of course, these PC based ESBTs are not serious contenders for a large scale expert system development. However, this evaluation does suggest that one approach to the selection of an ESBT is to gradually relax important constraints and criteria until the choice set contains clearly infeasible solutions. It is then possible to develop a clearer insight into the trade-offs that are being made as the boundary from feasible to infeasible is crossed. In the case of the TI evaluation of FRESH tools, one presumes that the additional cost of acquiring a copy of Nexpert (approximately $5,000 according to Gevarter) would not have been a significant item in the overall budget, and could have generated useful comparative information against which to assess the capabilities of the serious contenders. Conversely, the use of Nexpert prior to the full-scale evaluation might have have assisted in developing appropriate evaluation criteria, thereby making the overall process more effective and efficient.

3.2. AN ASSISTANT FOR SCIENCE AND TECHNOLOGY ANALYSIS

Our second example is concerned with the evaluation of a decision aid for Science and Technology analysts. The analyst's assistant system is intended to provide radar analysts with capabilities for inter-analyst communication, semiautomatic report production, information structuring and storage, interactive knowledge-based hypothesis formation in the field of radar, intelligent information retrieval from multiple databases, and access to analytical models.

A knowledge-based research prototype has been created for analyzing existing radar systems. This expert system, called ASTA (Cromarty *et al.*, 1986; and reprinted in Appendix A), interactively accepts values for system attributes, subsystem attributes, or signal parameters, and then incrementally infers the value of as many other attributes as it can. ASTA is tailored to radar systems by virtue of its knowledge base of facts and heuristics, derived from radar designers and analysts and from radar design handbooks, that defines how new information should be inferred from existing

Table 3-3a Alternative Evaluation of FRESH Tools

Attribute	KEE	ART	K-C	PICON	S1
Knowledge Representation					
Frames	*	*	*	*	*
Inheritance	*	*	*	*	
Demons and Procedures	*	*	*		*
Hypothetical Reasoning					
Truth Maintenance	*	*			
Context	*	*	*		
Reasoning Control					
Forward Chaining	*	*	*	*	*
Backward Chaining	*	*	*	*	*
Explanation Capability					
"Count"	3	3	3	2	3
Ease of Engineering					
"Count"	3	2	2	3	3

Table 3-3b Alternative Evaluation (cont.)

Attribute	ES Env.	Envisage	KES	M1	Nexpert
Knowledge Representation					
Frames					*
Inheritance					*
Demons and Procedures		*			*
Hypothetical Reasoning					
Truth Maintenance					*
Context					
Reasoning Control					
Forward Chaining	*	*		*	*
Backward Chaining	*	*	*	*	*
Explanation Capability					
"Count"	3	3	3	3	3
Ease of Engineering					
"Count"	1	1	1	1	3

Table 3-3c Alternative Evaluation (cont.)

Attribute	PC+	ExSys	ExpertEdge	ESP	Insight
Knowledge Representation					
Frames	*			*	
Inheritance	*			*	
Demons and Procedures	*			*	
Hypothetical Reasoning					
Truth Maintenance			*		
Context					
Reasoning Control					
Forward Chaining	*	*		*	*
Backward Chaining	*	*	*	*	*
Explanation Capability					
"Count"	3	3	3	3	3
Ease of Engineering					
"Count"	2	2	2	3	1

Table 3-3d Alternative Evaluation (cont.)

Attribute	TIMM	RuleMaster	KDS3	1stClass
Knowledge Representation				
Frames			*	
Inheritance			*	
Demons and Procedures				
Hypothetical Reasoning				
Truth Maintenance			*	
Context				
Reasoning Control				
Forward Chaining	*	*	*	*
Backward Chaining		*	*	*
Explanation Capability				
"Count"	1	3	3	2
Ease of Engineering				
"Count"	2	1	2	3

information. ASTA hypotheses are dynamic, undergoing constant revision both by analysts and from automatic data entry. Underlying these networks is a relatively static knowledge base that represents information about such things as types of hypotheses that are frequently created, a taxonomy of national and foreign radar and EW equipment, research organizations and personnel, development cycles, and rules for searching textual documents relevant to each radar concept. The combination of knowledge bases about radar and EW, an interactive environment for hypothesis formation, and automatic retrieval from multiple databases potentially provides an extremely powerful capability to science and technology analysts.

The intent of our example is to assess how well ASTA is meeting that potential.

3.2.1. Evaluation Attributes for the ASTA Domain

We have identified six groups of attributes that characterize the performance of the ASTA system. These are concerned with: (1) the properties of the domain data used by ASTA, (2) the way in which information is presented to the user, (3) the facilities available for explanation, documentation and help, (4) the system's ability to perform multiple forms of inference, (5) general systems-level issues, and (6) the facilities for accessing remote databases. We will discuss each of these in turn.

3.2.1.1. Domain Data Attributes

The input data to ASTA consists of reports on the various characteristics of observed radar signals and system attributes. The primary concern is, therefore, that all the relevant information available in these reports is actually encoded. That is, that all the interesting features of the signal are recorded so that a complete analysis is possible. A related concern is that the parameterization of the signal used as the basis for the reasoning that ASTA performs is minimal, general and will indeed support proper inference. Then thirdly, we are concerned that the internal representation of the data is efficient and has clarity and intuitiveness. These attributes are summarized in Figure 3-4.

3.2.1.2. Information Presentation Attributes

The second group of attributes is concerned with the quality of the displays provided within ASTA. The first issue is how well the display supports efficient data entry. The second is a question of the interactive "feel" of the interface. Under the latter, we have identified three major sub-attributes; how well the display supports the user in moving around within the system, how well the screen is laid out with respect to the cognitive expectations of the analyst engaged in a particular task, and whether the system provides the correct level of detail in response to a user request. These are summarized in Figure 3-5.

3.2.1.3. Explanation, Documentation and Help Attributes

The third group of attributes addresses the capability of the system to provide assistance to the user (especially the novice user) in understanding both the process of

1. Domain Data
   1.1 Input Data
      1.1.1 Sufficiency

   1.2 Parameterization
      1.2.1 Minimal
      1.2.2 General
      1.2.3 Support Proper Inference

   1.3 Representation
      1.3.1 Efficiency
      1.3.2 Clarity and Intuitiveness

Figure 3-4 Domain Data Attributes

1. Presentation
   1.1 Efficient Data Entry

   1.2 Interactive Feel
      1.2.1 Navigation
      1.2.2 Screen Layout
      1.2.3 Level of Detail Presented

Figure 3-5 Information Presentation Attributes

S&T analysis and the behavior of the ASTA system itself. Thus performance attributes capture such notions as how well the system educates novice users and helps extend domain understanding. They also capture the ability of the system to establish a common terminology for the users and the system, to support an understanding of the implementation of the system, to assist the user in the operation of the system, and to help the user understand the reasoning that the system performs. The ASTA system contains a number of "tools" such as introductory documentation, a tutorial text, an on-line bibliography, a glossary, and an on-line help system, that assist in providing these capabilities. The inter-relationships between these attributes is shown in Figure 3-6.

1. Explanation, Documentation and Help
    1.1 User Understanding of Analysis
        1.1.1 Educate Novice Users
            1.1.1.1 Introductory documentation
            1.1.1.2 Tutorial text
        1.1.2 Help Extend Domain Understanding
            1.1.2.1 Bibliography

    1.2 User Understanding of Expert System
        1.2.1 Educate Novice Users
            1.2.1.1 Introductory documentation
        1.2.2 Establish Common Terminology
            1.2.2.1 Glossary
        1.2.3 Support Understanding of ES Implementation
            1.2.3.1 Descriptions of internal data structures
            1.2.3.2 English description of rules
            1.2.3.3 Code versions of rules
        1.2.4 Assist User in Operation of ES
            1.2.4.1 On-line help
        1.2.5 Help User Understand ES Reasoning
            1.2.5.1 Explanation

Figure 3-6 Explanation, Documentation and Help Attributes

3.2.1.4. Inference Control Attributes

Another important group of attributes is concerned with inference control. Since the ASTA system is partly experimental, it is important that multiple inference paradigms are available and that their selection is under software control at runtime. In addition, it is important that the system support multiple data representations for the same rule-set. Another requirement is that the knowledge engineer have runtime access to the inference mechanisms so that the system can be "tuned". Finally, the system must support the construction of multiple simultaneous hypotheses and must be able to maintain consistency between the various groups of hypotheses. These attributes are illustrated in Figure 3-7.

3.2.1.5. System Attributes

The fifth set of issues relates to various system level questions. These are shown in Figure 3-8. Thus issues of concern are the portability of the code, the system's overall performance, the ease with which knowledge and software can be managed, and the overall modularity of the system design.

3.2.1.6. Database Search Attributes

The final group of attributes relate to ASTA's ability to access remote databases. There are two major questions here: does the system provide the kinds of access required by the user and the system itself, and is the performance adequate? This latter question is further decomposed into questions about the ability of ASTA to utilize the good DBMS technology implemented in the remote databases, the quality of the responses generated in answer to DB queries, and whether this remote access helps build-up the analyst's intuitions about the specific analysis problem under consideration as well as the domain itself. These concerns are illustrated in Figure 3-9.

3.2.2. ASTA Evaluation Frames

To construct evaluation frames we need to specify who is evaluating and why they are evaluating the system. So in Table 3-4 we list a number of possible evaluators and show their major areas of concern. This table illustrates that different evaluators will have different areas of concern. Thus the novice analyst is concerned only that the system has good presentation capabilities, provides good explanation, documentation and help (EDH in the column headings), and allows proper access to appropriate remote databases. In contrast, the expert analyst has additional concerns with the system's capabilities in representing the domain data correctly and has some interest (denoted (*) in the table) in the system's inference capabilities, but has less concern with the EDH capabilities. The expert has, like the novice, a concern with the remote database access capability of the system.

The analysts' manager has a completely different view however. He is not concerned with the task specific attributes, except insofar as the system is providing adequate support for his analysts. He is very concerned, though with the system issues and the remote access capabilities. That is, his concerns are with overall

1. Inference Control
   1.1 Flexibility and Breadth
      1.1.1 Multiple Inference Paradigms
      1.1.2 Software Control of Paradigm Selection
      1.1.3 Multiple Data Representations

   1.2 Interactiveness
      1.2.1 Runtime Access for Knowledge Engineer

   1.3 Multiple Hypotheses
      1.3.1 Multiple Simultaneous Hypotheses
      1.3.2 Consistency Management

Figure 3-7 Inference Control Attributes

1. System Issues
   1.1 Portability
      1.1.1 Software Portable to Other CPUs
      1.1.2 Interface S/W Portable to Other Display Devices

   1.2 High Performance
      1.2.1 Speed
      1.2.2 Reliability

   1.3 Knowledge Management
      1.3.1 Ease of Maintenance
      1.3.2 Ease of Extension

   1.4 Software Management
      1.4.1 Ease of Maintenance
      1.4.2 Ease of Extension

   1.5 Modularity of Design
      1.5.1 Clean Model of Program

Figure 3-8 System Attributes

1. Database Search
   1.1 DB Location and Access
      1.1.1 User Access to Data
      1.1.2 ES Access to Data

   1.2 Performance
      1.2.1 Utilize Good DBMS Technology
      1.2.2 Effective Answers to DB Queries
      1.2.3 Help Build Analysts Intuitions

Figure 3-9 Database Search Attributes

system performance and maintainability, and with the system's ability to interface to other computer-based tools that might be available in the analysts' environment.

Two other evaluators are the system developer (or system maintainer) who has concerns in every attribute category, and the domain expert who's knowledge in encoded in the system. The domain expert is not at all concerned with the system and database issues, but is concerned with the task specific attributes.

It is beyond the scope of this effort to perform a complete evaluation using all these evaluators. However, we see that each might actually produce different evaluations since they have different concerns and gives different weights to the various groups of attributes. In the next section we will focus on the system developer and generate an evaluation for three different evaluation questions.

Table 3-4 The Evaluator-Attribute Relation

	Data	Presentation	EDH	Inference	System Issues	DB Search
Novice Analyst		*	*			*
Expert Analyst	*	*	(*)	(*)		*
Analysts Manager					*	*
System Developer	*	*	*	*	*	*
Domain Expert	*	*	*	(*)		

3.2.3. The System Developer's Evaluation Frame

The system developer is concerned with all aspects of the ASTA system, and therefore needs to construct an evaluation for all attributes. To do this we address three evaluation questions. These are:

- How good was the design for ASTA?
- How good is the current implementation?
- How good will it be with planned improvements?

Given that ASTA is still a prototype system, the answers to these questions will be highly subjective. Accordingly, we chose a simple six-point measurement scale on which to record the evaluations (ratings range from A to F; A being the best and F the worst). We also found that it was unnecessary to define formal combination functions for ratings. The system developer was completely at ease with subjective

combinations.

Tables 3-5a and 3-5b show the ratings for all the attributes for the three questions. First of all, we note that, in general, the ratings given by the system developer are at least a "C". That is, the design, implementation and planned developments all are more than adequate in meeting the needs and requirements of the S&T domain. The only attribute for which this is not the case is attribute 3.2.2, "Support Understanding of ES Implementation". The ratings D,E,E indicate that even the original design was deficient in its ability to support this need, and that future effort will not attempt to address it. Actually, this simply reflects a change in the requirements that has taken place since the original system design. Once the first implementations of the system were available, it was recognized that it was not in fact an important goal that the user become familiar with the implementation of the expert system. What was really required was that the user should understand the basic operation of the system, together with its underlying principles, and not the specific details of the software. This is reflected in the overall rating for attribute 3.2, *User Understanding of Expert System*, where the poor rating for attribute 3.2.3 is essentially ignored.

A second general point is that the ratings are usually ordered so that the rating for the design is higher than or equal to the rating for the future system, which in turn is higher than the rating for the current implementation of the system. This ordering simply reflects the fact that the ASTA system is still in the relatively early phases of its planned development cycle. The development effort to date has concentrated on the implementation of a broadly based capability, and future efforts will complete the implementation of the planned capability. However, in a number of interesting cases the planned future performance is expected to be superior to the original design. We see that attributes 1.1, "input data", and 1.2, "parameterization", both show the future rating to be "A" when the design only receives a "B". This increase reflects both the fact that capturing the input data properly is perhaps the most critical part of the system, and that as the developer's experience with the system has increased a greater understanding of the issues has arisen. This makes it possible to improve upon the original design and to concentrate future efforts in this area.

3.3. A SYSTEM FOR FULL-TEXT DOCUMENT RETRIEVAL

Our third example is also an instance of a decision aid evaluation, and is based on some evaluation studies currently being performed with the RUBRIC full-text document retrieval system (Tong *et al.*, 1985; and reprinted in Appendix B). This system is designed to provide the intelligence analyst with tools for constructing knowledge-based descriptions of documents she would like retrieved from very large collections of full-text documents. Once documents have been retrieved they are used as the basis for the analysis being developed. Thus, unlike the ASTA system which we might think of as an "assistant analyst," the RUBRIC system is actually an adjunct to the analysis process.

Table 3-5a System Developer's Evaluations

Attribute	Rating		
	Design	Implemented	Future
1. **Domain Data**	B	B	B
1.1 *Input Data*	B	B	A
1.1.1 Sufficiency	B	B	A
1.2 *Parameterization*	B	B	A
1.2.1 Minimal	B	B	A
1.2.2 General	B	C	A
1.2.3 Support Proper Inference	A	B	A
1.3 *Representation*	B	B	B
1.3.1 Efficiency	B	C	C
1.3.2 Clarity and Intuitiveness	B	C	B
2. **Presentation**	A	B	A
2.1 *Efficient Data Entry*	A	B	A
2.2 *Interactive Feel*	A	B	A
2.2.1 Navigation	A	B	B
2.2.2 Screen Layout	A	B	B
2.2.3 Level of Detail Presented	A	A	A
3. **Explanation, Documentation and Help**	A	C	B
3.1 *User Understanding of Analysis*	A	C	B
3.1.1 Educate Novice Users	B	C	B
3.1.2 Help Extend Domain Understanding	A	C	B
3.2 *User Understanding of Expert System*	B	C	B
3.2.1 Educate Novice Users	B	C	B
3.2.2 Establish Common Terminology	A	C	B
3.2.3 Support Understanding of ES Implementation	D	E	E
3.2.4 Assist User in Operation of ES	B	C	B
3.2.5 Help User Understand ES Reasoning	A	C	B

Rating Scale: A - F (A = highest rating, F = lowest rating)

Table 3-5b System Developer's Evaluations (cont.)

Attribute	Rating		
	Design	Implemented	Future
4. Inference Control	B	B	B
4.1 *Flexibility and Breadth*	B	B	B
4.1.1 Multiple Inference Paradigms	A	B	A
4.1.2 Software Control of Paradigm Selection	A	B	A
4.1.3 Multiple Data Representations	B	C	B
4.2 *Interactiveness*	B	B	B
4.2.1 Runtime Access for Knowledge Engineer	B	B	B
4.3 *Multiple Hypotheses*	A	B	A
4.3.1 Multiple Simultaneous Hypotheses	A	B	A
4.3.2 Consistency Management	A	B	A
5. System Issues	A	B	A
5.1 *Portability*	A	A	B
5.1.1 Software Portable to Other CPUs	A	A	A
5.1.2 Interface S/W Portable to Other Display Devices	A	A	B
5.2 *High Performance*	B	C	B
5.2.1 Speed	B	C	B
5.2.2 Reliability	B	C	B
5.3 *Knowledge Management*	B	B	A
5.3.1 Ease of Maintenance	B	B	A
5.3.2 Ease of Extension	B	B	B
5.4 *Software Management*	B	B	A
5.4.1 Ease of Maintenance	B	B	B
5.4.2 Ease of Extension	A	B	A
5.5 *Modularity of Design*	A	A	A
5.5.1 Clean Model of Program	A	A	A
6. Database Search	A	C	A
6.1 *DB Location and Access*	A	C	A
6.1.1 User Access to Data	B	C	A
6.1.2 ES Access to Data	A	C	A
6.2 *Performance*	A	C	A
6.2.1 Utilize Good DBMS Technology	B	C	B
6.2.2 Effective Answers to DB Queries	A	B	A
6.2.3 Help Build Analysts Intuitions	A	C	A

Rating Scale: A - F (A = highest rating, F = lowest rating)

In this example we have a situation in which the analyst needs to produce a number of different types of intelligence product. The first is a weekly paper on trends and significant activities in their several areas of expertise, the second is an in-depth retrospective analysis of a particular analysis question, and the third is an executive summary of key issues generated in response to a high level request. These tasks are currently supported by a number of database resources (including a conventionally indexed hard-copy archive and on-line access to some material of interest), by a stand-alone word processing system, and by the analyst's own personal library of documents and notes.

By providing the analysts with an advanced tool for information retrieval we would expect both their efficiency and effectiveness to increase. The goal of this example is to see how we might construct evaluation frames that assist us in testing this hypothesis.

3.3.1. Performance Attributes

The first step in our evaluation process is to define the space of performance attributes that will apply to this situation. In our investigation we have identified five major groups of attributes: (1) those that are concerned with the information retrieval task and its relationship to analysis, (2) those that relate to the user interface, (3) those that relate to the integration of RUBRIC with existing systems, (4) those that relate to organizational considerations, and (5) those that relate to systems support questions. Our discussion will consider each group in turn.

3.3.1.1. Problem Attributes

This group of attributes relate to the requirements specific to the problem faced by the analyst. That is, they relate to the actual task of retrieving documents and using them in analysis. The attribute structure is summarized in Figure 3-10 where, as before, we use an indented list format to illustrate the hierarchical structure.

Notice that there are two main sub-groups, labelled "Information Retrieval" and "Analysis," which are concerned with the two main functions that RUBRIC is designed to support. Under Information Retrieval there is a further subdivision into attributes that are concerned with the formulation of queries and those that are concerned with the execution of queries. So, the expressiveness of the query language and the ease with which complex queries can be constructed are distinct but related concerns that impact query formulation. Similarly, precision, recall and speed of response are attributes that impact query execution.

Given that at this stage in the project the RUBRIC system is expected to give only minimal support for actual analysis, the attributes under Analysis are simply a list of basic functions that the system should support. Statistical analysis refers to a capability for generating statistical summaries of the retrieval results. Discriminant analysis refers to a capability for partitioning and sorting retrievals based on characteristics such as the document source or its date. Finally, detection of text differences refers to a technique for detecting documents that differ in specific ways

1. Problem Requirements
    1.1 Information Retrieval
        1.1.1 Query Formulation
            1.1.1.1 Expressiveness of query language
            1.1.1.2 Ease of knowledge engineering
        1.1.2 Query Execution
            1.1.2.1 Precision of retrievals
            1.1.2.2 Recall of retrievals
            1.1.2.3 System response time

    1.2 Analysis Support
        1.2.1 Statistical Analysis of Results
        1.2.2 Discriminant Analysis
        1.2.3 Detection of Text Differences

Figure 3-10 Problem Attributes

from expected patterns. In each case, the functionality provided by RUBRIC is intended to be at the "proof of concept" level.

As with the other examples we have considered, the definition of attributes is a subjective exercise. Although there are some structures that we would expect to generate little argument (e.g., that the key attributes of query execution are precision, recall and response time), others such as expressiveness and ease of engineering have no such clear-cut definition or inter-relationship.

3.3.1.2. User Interface Attributes

Attributes in this group relate to the characteristics of the user interface provided by RUBRIC. Figure 3-11 summarizes the main attributes and shows their interactions. They are grouped under three main categories; attributes relating to the

1. User Interface
   1.1 Input
      1.1.1 Rule Entry and Editing
      1.1.2 Query Entry

   1.2 Control
      1.2.1 Navigation
      1.2.2 Operation Selection
      1.2.3 Parameter Selection

   1.3 Output
      1.3.1 Knowledge Display
      1.3.2 Results Display
      1.3.3 General Information Display

Figure 3-11 User Interface Attributes

input of knowledge and queries, attributes relating to the control of and the movement around in the system, and attributes relating to the output of information. Under each of the main headings we have listed a few specific attributes that define how the main attributes should be interpreted in the context of RUBRIC.

So under Input, we are concerned with the facilities provided for the entry and editing of rules (rules are the basic knowledge representation mechanism in RUBRIC) and with the facilities provided for the entry of queries. Under Control, we are concerned with the ease with which the user can move around in the system, can select various operations to be performed, and can set various system parameters and variables (e.g., selecting the databases to be searched). Finally, under Output, we are concerned with facilities provided for the display of knowledge in the knowledge base, for the display of retrieval results, and for the display of general system information

(including system status information and on-line help).

Notice that although RUBRIC is a decision-aid in the same sense as ASTA, the interface concerns are structured differently (although they do of course have some commonality). This is a result of the style of interaction and the purposes of the two systems. In RUBRIC the analyst is acting as her own knowledge engineer and has total control over the actions of the system. This implies an increased level of concern with inputs and control activities than we see in ASTA.

3.3.1.3. System Integration Attributes

Since RUBRIC is intended to be just one of many automated tools that the analyst can use, there are a group of attributes that relate to the degree of integration between RUBRIC and these other capabilities. Figure 3-12 shows the three major categories of concerns. Given the preliminary stage of the RUBRIC development, the further refinement of these attributes is unnecessary. However, the general sense of them is that RUBRIC must be able to access, and receive as input, data from external systems (e.g., remote databases), must provide capabilities for accessing other analysis tools without having to leave RUBRIC (i.e., the tools should be embedded in RUBRIC), and must have the capability for sending its output (typically document fragments) to other systems (e.g., word processors and personalized DBMSs).

3.3.1.4. System Support Attributes

As with any computer-based tool, there are concerns about the quality of the vendor support for the tool once it has been installed at the user site. Again, since RUBRIC is at an early stage of development, the details of these attributes are to be determined, but can be summarized as in Figure 3-13. The issues of concern are the amount and quality of the training and documentation, the level of disruption caused by the installation of the system, and the level and timeliness of the on-going support both for system maintenance and knowledge engineering.

3.3.1.5. Organizational Attributes

The final group of attributes are concerned with some of the more intangible organizational aspects of introducing advanced computer based capabilities. These are summarized in Figure 3-14. The intent here is capture some subjective impressions of how the RUBRIC system integrates with existing strategic plans for modernization of the organization's computer facilities (both in terms of its functionality and hardware/software configuration). In addition, we are concerned with how it affects the existing working environments of the user and the ADP staff. That is, we are concerned with how accessible the system will be to the user (e.g., on her desk versus down the hall) and whether the ADP staff have familiarity with the hardware and software so that they may perform basic system support themselves. Then thirdly, we are interested in its impact on the sociological structures within the organization. So for example, we might be concerned with the analyst's reactions to computer based tools in general, management's reaction to the new capability and its potential uses in

1. Integration with Other Systems
   1.1 Input of External Data
   1.2 Access to On-Line Tools
   1.3 Output to External Systems

Figure 3-12 System Integration Attributes

1. System Support
   1.1 Training and Documentation
   1.2 Installation
   1.3 On-Going Support
      1.3.1 System Maintenance
      1.3.2 Knowledge Engineering Support

Figure 3-13 System Support Attributes

1. Organizational Integration
   1.1 Integration with Strategic Plans
   1.2 Effect on Working Environment
      1.2.1 Effect on Users
      1.2.2 Effect on ADP Staff
   1.3 Effect on Organizational Sociology

Figure 3-14 Organizational Attributes

facilitating analysis production, and ways in which the system might facilitate the sharing of analysis and analysis techniques amongst the community of users.

3.3.2. The Analyst's Evaluation Frame

The first evaluation frame that we will construct is based on the analyst's interaction with the system. Some of the contextual information that attaches to the frame is that the current system is an early field prototype with intentionally limited functionality in certain areas, and that the few analysts selected to perform the evaluations have to date had only limited opportunity to work with the RUBRIC system as installed at their site.

The first step in constructing the frame is to select from the attribute space those attributes that are relevant to the context that the frame defines. Clearly, the problem and user interface attribute structures are required in their entirety. However, since

the current version of the system is not intended to be integrated with other tools, nor is it intended to be physically connected to remote systems, not all the system integration attributes seem to be required. That is, only the input and output attributes are used, and actually have a restricted interpretation. All of the system support attributes are relevant, but, given the perspective of the analyst and the fact that the RUBRIC system is a prototype, the organizational attributes have marginal relevance and can be safely ignored without affecting the evaluation. The frame now contains an instantiated version of the attributes copied from the attribute space, with the various fragments connected together as shown in Figure 3-15.

The next step is, in general, to attach measurement procedures to each of the attributes. For the terminal attributes this will be a procedure for acquiring the necessary data, but for the intermediate attributes this may simply be a function for combining the values from its child attributes. Following that, it remains to take the measurements and then propagate the values through the attribute structure.

The details of this process will obviously depend on the types of measurements made and the model of preferences used. If we were to select a multi-attribute model then it would be necessary to define utility functions for each of the attributes in terms of its descendants. If we selected a linguistic variable model then we would need to calibrate the linguistic variables and construct the rule-based combining functions.

In this example though, we choose to make use of an even weaker evaluation method. We simply allowed the analyst to provide subjective judgements on a seven-point scale; a value of one is the lowest value, seven the highest, and four the mid-point. Combination of values was also done by the analyst, although through discussion of the resulting values it was usually possible to infer her preferences and simple numerical combining functions. The resulting evaluation is shown in Table 3-6.

In general, the analyst had little difficulty assigning scores to the attributes. Where difficulty did occur is was usually because the attribute was very general in character or was vaguely defined (e.g., the attribute labelled Analysis Support; level 1.2 in Figure 3-10). The evaluation generally proceeded by assigning a rating to an attribute one level above the terminals, then assigning a rating to the terminals themselves and finally adjusting the rating for the higher level attribute if necessary. This process was then repeated at one level higher in the attribute structure to generate the next level of ratings.

The ratings in Table 3-6 exhibit a number of interesting features. First of all, we see that the overall ratings for the main attribute groups are from good to very good (i.e., numerical ratings of 5 or 6) despite some average and below average ratings for some of the sub-attributes (i.e., ratings of 4 or less). The reason for this is that those attributes with low ratings are either not considered particularly important at this stage of the RUBRIC system development, or they are offset by high ratings on other attributes. So, for example, the low rating for attribute 1.1.1.2, Ease of Knowledge

1. Problem Requirements
   1.1 Information Retrieval
      1.1.1 Query Formulation
         1.1.1.1 Expressiveness of query language
         1.1.1.2 Ease of knowledge engineering
      1.1.2 Query Execution
         1.1.2.1 Precision of retrievals
         1.1.2.2 Recall of retrievals
         1.1.2.3 System response time
   1.2 Analysis Support
      1.2.1 Statistical Analysis of Results
      1.2.2 Discriminant Analysis
      1.2.3 Detection of Text Differences

2. User Interface
   2.1 Input
      2.1.1 Rule Entry and Editing
      2.1.2 Query Entry
   2.2 Control
      2.2.1 Navigation
      2.2.2 Operation Selection
      2.2.3 Parameter Selection
   2.3 Output
      2.3.1 Knowledge Display
      2.3.2 Results Display
      2.3.3 General Information Display

3. Integration with Other Systems
   3.1 Input of External Data
   3.2 Output to External Systems

4. Systems Support
   4.1 Training and Documentation
   4.2 Installation
   4.3 On-Going Support
      4.3.1 System Maintenance
      4.3.2 Knowledge Engineering Support

Figure 3-15 Attribute Structure for the Analyst's Frame

Table 3-6 Analyst's Evaluation

Attribute	Rating
1. Problem Requirements	5
1.1 *Information Retrieval*	5
1.1.1 Query Formulation	5
1.1.1.1 Expressiveness of query language	7
1.1.1.2 Ease of knowledge engineering	3
1.1.2 Query Execution	6
1.1.2.1 Precision of retrievals	6
1.1.2.2 Recall of retrievals	6
1.1.2.3 System response time	4
1.2 *Analysis Support*	4
1.2.1 Statistical Analysis of Results	4
1.2.2 Discriminant Analysis	4
1.2.3 Detection of Text Differences	4
2. User Interface	5
2.1 *Input*	4
2.1.1 Rule Entry and Editing	3
2.1.2 Query Entry	6
2.2 *Control*	5
2.2.1 Navigation	4
2.2.2 Operation Selection	4
2.2.3 Parameter Selection	7
2.3 *Output*	7
2.3.1 Knowledge Display	6
2.3.2 Results Display	7
2.3.3 General Information Display	7
3. Integration with Other Systems	6
3.1 *Input of External Data*	6
3.2 *Output to External Systems*	7
4. Systems Support	6
4.1 *Training and Documentation*	6
4.2 *Installation*	7
4.3 *On-Going Support*	7
4.3.1 System Maintenance	7
4.3.2 Knowledge Engineering Support	7

Rating Scale: 1 - 7 (7 = highest rating, 1 = lowest rating)

Engineering, is balanced by a very high rating for attribute 1.1.1.1, Expressiveness of Query Language. That is, the analyst is willing to trade-off some effort in building knowledge-based queries for the ability to express exactly the retrievals needed. On the other hand, the average rating for attribute 1.2, Analysis Support, does not significantly effect the overall rating since the current version of the system was designed to have minimal support for the analysis functions.

Another interesting feature of the table is that the difficulty that the analyst has in building conceptually complex queries is reflected in the rating for attribute 2.1.1, Rule Entry and Editing. Thus, not only does the analyst see this a difficult task, but she also finds that the interface tools provided to support it are also rather poor. Of course, there is a strong dependency between these two attributes. Simple tasks generally require simple displays; complex tasks require more complex displays.

One final note on the table is that the analyst was satisfied to leave the ratings uncombined at the highest level. We take this to mean that no additional insights would be gained by aggregating the four main attributes, and that on the contrary, information would be lost if only the single aggregate score were produced.

3.3.3. The System Developer's Evaluation Frame

The second evaluation frame that we will consider is one that reflects the concerns of the system developer. In developing this frame, we discovered that the attributes described in Section 3.3.1 needed to be augmented with two groups of attributes that reflect the developer's consideration of the system with respect to the state of the art, and with respect to its general flexibility. The attributes for these two groups are summarized in Figures 3-16 and 3-17.

Under State of the Art, the developer considered five different issues. First, whether the hardware was advanced with respect to recent advances in computer science. Second, whether the software environment was similarly advanced. Third, whether the general level of technology incorporated into the system was high. Fourth, whether the knowledge representation reflected current thinking in AI. And fifth, whether the system *qua* system is sophisticated with respect to other information retrieval systems and with respect to other systems that the developer has been involved with.

Under Flexibility, the developer considered whether the system was easily tailorable to the needs of the analyst, was adaptable to potential changes in the definition of the tasks it was designed to support, was easily extensible to include new functionality, and whether the software was reusable by virtue of its being modular and portable.

The attribute structure of the frame is shown in Figure 3-18. Notice that some of the attributes are copied from the original attribute space (e.g., 3.1 System Integration, and 3.2 Organizational Integration) but that those under System Performance are actually new attributes even though they are similar to those used in the analyst's frame. Thus Response Time is intended to be an overall assessment of

1. State of the Art
   1.1 Hardware
   1.2 Software
   1.3 Technology
   1.4 Knowledge Representation
   1.5 Systems

Figure 3-16 State of the Art Attributes

1. System Flexibility
   1.1 Tailorability
   1.2 Adaptability
   1.3 Extensibility
   1.4 Reusability

Figure 3-17 Flexibility Attributes

1. System Performance
   1.1 Response Time
   1.2 Validity
   1.3 Reliability

2. State of the Art
   2.1 Hardware
   2.2 Software
   2.3 Technology
   2.4 Knowledge Representation
   2.5 Systems

3. System Compatibility
   3.1 Integration with Other Systems
   3.2 Organizational Integration

4. System Flexibility
   4.1 Tailorability
   4.2 Adaptability
   4.3 Extensibility
   4.4 Reusability

Figure 3-18 System Developer's Evaluation Frame

the system's ability to respond to user requests, not just the time needed to perform a retrieval. Similarly, whilst Validity includes Precision and Recall, it also includes a wider concept of a match between the system's overall capabilities and those needed by the analyst. Reliability, of course, relates to the system's overall reliability when in continuous use.

The developer generated the evaluations shown in Table 3-7, again using a seven point measurement scale and a subjective combination scheme. Generally, the evaluations are better than average although we see that with respect to the state of the art (attribute 2), the overall rating is only average. The reasons for this overall rating are that the general level of technology (in particular, AI technology) and the overall systems capability are just average, whilst, in addition, the knowledge representation used would probably be considered "passe" in AI research communities. Note, however, that being only average in the state of the art may be a distinct *advantage* when it comes to assessing the overall suitability of the system. The customer may well resist high technology solutions to her problem of the grounds that they are high risk solutions. We might thus expect to see an inverse correlation between the rating on state of the art and the ratings on system compatibility, system reliability and system flexibility.

3.3.4. The Question of Overall Utility

The evaluations produced in the previous two sections represent evaluations of the RUBRIC system as an advanced information retrieval and analysis tool, but do not directly address the question of the overall utility of the system. That is, they measure how well RUBRIC does with respect to the goals and requirements set for it at this stage in the development cycle, but they do not address the value of RUBRIC as a part of the analyst's overall task environment.

To perform this kind of evaluation we have to construct an attribute structure that reflects the analyst's concerns with her main daily activity. That is, we need to focus on the analysis product itself. In Figure 3-19 we show a simple structure that reflects some high level concerns. So, the analyst is concerned with the quality of the analysis she produces in terms of its comprehensiveness, its conciseness and her of level of confidence in the final product. Similarly, she is concerned with the amount of time and level of effort it takes to produce the analysis. Then finally, she is concerned with the impact new tools have on her ability to develop advanced analytic skills and to become more familiar with the various resources available to her.

To properly assess the impact of RUBRIC it would be necessary to develop a more detailed model of the analyst's working environment. This would need to include a description of the ways in which information retrieval as an activity interacts with other activities such as reading, writing and conversations with colleagues. Similarly, the model would also need a description of the cognitive processes employed in going from textual data to a final analysis. Both of these are outside the scope of the current project, and so we leave the reader to imagine how we might construct an evaluation frame for the overall utility of the RUBRIC system.

Table 3-7 System Developer's Evaluations

Attribute	Rating
1. System Performance	5
1.1 Response Time	4
1.2 Validity	5
1.3 Reliability	6
2. State of the Art	4
2.1 Hardware	5
2.2 Software	5
2.3 Technology	4
2.4 Knowledge Representation	3
2.5 Systems	4
3. System Compatibility	6
3.1 Integration with Other Systems	6
3.2 Organizational Integration	7
4. System Flexibility	6
4.1 Tailorability	6
4.2 Adaptability	6
4.3 Extensibility	7
4.4 Reusability	5

Rating Scale: 1 - 7 (7 = highest rating, 1 = lowest rating)

1. Overall Utility
   1.1 Quality of Analysis
      1.1.1 Comprehensiveness
      1.1.2 Conciseness
      1.1.3 Confidence in the Results

   1.2 Efficiency of Analysis
      1.2.1 Time to Complete
      1.2.2 Level of Effort

   1.3 Expertise Development
      1.3.1 Resources Available
      1.3.2 Analysis Techniques

Figure 3-19 Overall Utility Attributes

3.4. COMMENTARY

The examples discussed above serve both to illustrate and test the workings of our proposed approach. They show that over a range of evaluation situations it is possible to create attribute spaces and evaluation frames which accurately reflect the concerns of evaluators. Furthermore, they show that the different frames can produce different evaluations, and thus confirm the advantage of performing a proper structuring of the evaluation knowledge. Indeed, we might argue that most of the benefit derived from the evaluation comes from this structuring exercise.

Of course, the examples are somewhat limited in that they address only two of our classes of AISs and make use of highly subjective interpretation methods supplied by ADS staff. Nevertheless, we can view them as "pencil and paper" pilot experiments that help us define the needs and requirements for a prototype evaluation environment. An immediate need is for a collection of tools that assist in the

construction of evaluation structures and the development of actual evaluations. Since we see the evaluation process as being iterative and highly dependent upon the evaluator, the environment needs to be very flexible and very interactive. We address the design of such an environment in the next chapter.

Chapter 4

AN EVALUATION ENVIRONMENT

In this chapter we describe a prototype system that can support both the evaluation of AISs, and research and development into evaluation methods. That is, we conceive of this system as an environment in which to explore the major issues in performance evaluation as well as one in which to perform actual evaluations. The system is organized around the methodological approach described in Chapter 2 and embodies some of the capabilities discussed in Chapter 3.

In the first part of the chapter we discuss a "workbench" approach to the design of the environment, in the second part we describe the top-level system architecture, and in the third we describe the main tools supported by the environment. Then finally, we consider some of the development issues associated with implementing such a system.

4.1. THE WORKBENCH APPROACH

As its name implies, a workbench approach to the design of the evaluation environment emphasizes the need to explore ways of doing evaluations, as well as doing the evaluations themselves. Thus the workbench should be able to support multiple approaches to evaluation and should allow the "user" to experiment with different evaluation techniques. Notice that our definition of a user in this situation is different from that employed in Chapter 3. Here the user is the evaluator or experimenter. Thus, as with the conventional workbench, a main feature of this environment is the availability of a large number of tools that can be used to assist in the development of evaluation methods and the evaluation of systems.

The main premise behind adopting this approach is that whilst there are a number of formal techniques for performing multi-attribute evaluations, there is little practical knowledge about the overall process of evaluation and its relationship to decision-making. The goal of the environment is, therefore, to help develop a body of such knowledge as it relates to the evaluation of AISs. Since this knowledge can only emerge through a series of experiments with various techniques and a range of AISs, it is crucial that we provide a flexible environment for performing these experiments.

The primary characteristic of this environment is, as we have already suggested, that it contains a large selection of tools. We shall return to these in more detail in Section 4.3 but, briefly, we can distinguish between tools for knowledge acquisition, tools for evaluation, and tools to support exploration. These tools have access to common data so that work with one can be suspended whilst another tool is used. Since we expect that the tools themselves might become quite complex, we envisage that the environment will itself have knowledge about tools and their uses so that the experimenter can be given advice about, and help with, their selection and utilization.

4.2. A SYSTEM SPECIFICATION

Since we need to provide a flexible, extensible and easy to use environment, the top-level requirements for the system are relatively straightforward. Overall activity is managed by an *Environment Manager* which acts to connect the user to the appropriate tools. This is shown in Figure 4-1. We see that there are a number of major elements in this design and, apart from the *Toolbox* which is described in the following section, we will consider them in turn.

4.2.1. The Knowledge Base Management System

After the toolbox, the most important element of the design is the *Knowledge Base Management System*. It is within this sub-system that all the knowledge used by the tools is stored. Within the knowledge base are a number of distinct bodies of knowledge that must be managed to support efficient and effective access by the tools.

First, there is the *Attribute Library*. This is the repository for knowledge about evaluation attributes. The information stored along with the attribute name, includes the purpose of the attribute, the names of possible or preferred measurement procedures, the range of values the attribute can take, and the names of any compound attributes that can be derived from it.

Second, there is the *Frame Library*. This properly consists of two sub-libraries: a library of standard or prototypical frames that might be used in an evaluation, and a library of user frames that the experimenter may want to use in subsequent evaluations. The prototypical frames might be thought of as "standard evaluation contexts" that over time have become accepted as appropriate for certain evaluation problems. The user frames, on the other hand, represent either very specific evaluation contexts or experimental contexts. This library also contains perspectives, distinguished along similar lines.

The third library is the *Model Library*. This contains standard system behavior models (if they exist) that can be used to augment the development of evaluation frames. To this extent models can be thought of as compound attributes, although they will generally be much more complex and will have associated with them the idea of being executable. The model library also includes models of organizations. That is, models that can assist with understanding the relationships between evaluation frames when the frames reflect the context of individuals (or groups of individuals) within an organization. Such models would obviously be useful in constructing and interpreting perspectives.

A fourth library is the *Measurement Procedure Library*. Here is stored information about measurement procedures for attributes. This information would include alternative procedures, their associated costs, and the quality of the measurements produced by each of them. The procedures might be general or specific. That is they may address general purpose attributes such as response time, or cost, in which case the actual procedure can be described without direct reference to the system being evaluated. Or, they may address specific attributes such as the retrieval effectiveness

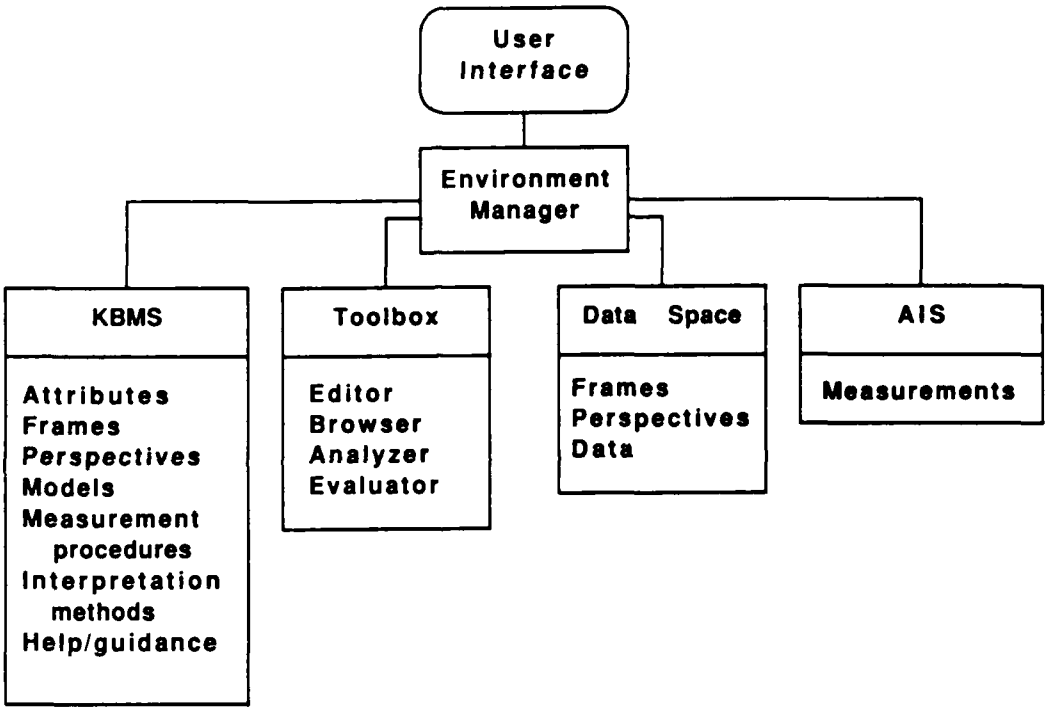


Figure 4-1 System Specification

in which the procedure must specify certain system specific information.

The fifth knowledge collection is the *Interpretation Method Library.* Here we store the information about various general methods for performing evaluations, the algorithms needed to compute evaluations from measurements, and the procedures for combining evaluations produced by different techniques.

The final knowledge collection is the *Help and Guidance Library.* In an advanced version of the environment, this would contain knowledge about the uses of the tools in the toolbox, the appropriateness of various evaluation procedures for specific problem types, suggestions for selecting and constructing evaluation frames, and, of course, general system level help.

4.2.2. The Evaluation Data Space

Whilst an evaluation is in progress, many frames, perspectives and other general data objects will be active. In our design, these objects are located in the evaluation data space. Examples would be partially completed frames, measurement data, and the evaluations themselves.

4.2.3. The AIS Under Test

At least conceptually, the AIS under test is under the control of the environment manager. Although we can certainly imagine a system that automatically performs measurements of the AIS, in the short term the AIS will be physically separate from the evaluation environment. The measurement procedures will thus be instructions on how to perform the actual measurements and how to report them.

4.2.4. The User Interface

Since we consider the process of evaluation to be a highly exploratory endeavour, the user interface needs to be highly interactive and graphically oriented. It should have a high resolution display to enable the graphical representation of frames and perspectives. It should have a large screen area so that multiple pieces of information can be displayed concurrently, and it should make use of color to add structure and additional semantics to the visually displayed information.

4.3. THE TOOLBOX

The toolbox is the heart of the evaluation environment. Although we can imagine a large variety of tools, we shall consider four generic ones. These cover the four major tasks performed using the environment, namely, editing and creating frames and perspectives, browsing the knowledge bases, analyzing the results of evaluations, and performing the actual evaluations.

4.3.1. The Editor

The editor is the tool used to create and modify frames and perspectives, as well as all the other objects under user control in the environment (e.g., attributes, measurement procedures, interpretation methods). Clearly, this task involves

extensive interaction with the evaluation data space and the KBMS, as well the co-operative use of other tools.

We can identify a number of separate sub-tasks that further define the editing activity, and the first of these is what we may call *Composition*. This is the task of actually joining together frames and perspectives. Generally, this will involve selecting elements from the knowledge bases followed by connecting them to other elements on the frame or perspective. Obviously, browsing of the knowledge base will often be a precursor activity. Another element of composition is providing capabilities to determine whether the proposed addition to the frame is "consistent" with the already existing knowledge. Unfortunately, consistency is not a straightforward concept. Checking that the new knowledge does not conflict with existing knowledge is complicated by the fact that there are often implicit constraints between the knowledge fragments of which the editor is unaware. Under these circumstances, the most useful behavior that the editor can exhibit is one of simply alerting the user to potential problems. Similarly, the editor needs to provide facilities for checking that the composing operations are syntactically correct, and facilities for prompting the user for the necessary information needed to complete the frame or perspective.

The second sub-task is *Modification*. This involves making changes and extensions to existing objects. This is obviously very similar to the basic composition activity, and the editor requirements are essentially the same. Additional capability that the editor may require is to be able to inform the user of the consequences of the proposed changes. For example, the user may wish to ask a number of *What if?* types of question; What if I add a new piece of knowledge, What if I delete a piece of knowledge, What would happen if I triggered this piece of knowledge, What if I change the preference value attached to this knowledge fragment?

The third sub-task is *Saving*. At some point it will be necessary to save objects either as complete entities to be permanently stored in the KBMS or as partially completed entities to be accessed later for further editing.

A fourth sub-task is *Annotation*. This task involves the provision of English language descriptions of the frames and the reasons for creating them. The descriptions could include extensive notes on the assumptions and rationales for selecting particular measurement procedures, interpretation methods and preferences. We imagine that these annotations will be primarily useful during browsing as a way of helping the user select elements from the various libraries.

Obviously, the editing task is a complex and highly context dependent activity. The display requirements would seem to be that the user be provided with a number of simultaneous views of the frame and its relationship to other objects. We would expect that a graphical representation of the evaluation structure would be most natural with nodes being attributes and arcs being dependencies. Additional information might be provided as text or tables, and the ability to graphically select elements in the graph, expand them and edit them is obviously an important requirement.

4.3.2. The Browser

The browser tool allows the user to explore the contents of the knowledge bases in the environment. Given that there are several forms of knowledge we would expect there to be several types of browsing needs. For example, browsing a collection of frames will demand rather different capabilities from browsing a collection of measurement procedures. Nevertheless, we can identify three broad classes of activity that reflect the user's requests for answers to the following questions: (1) What does the system know about? (2) Does the system know about a particular item? and (3) What specifically does the system know about that item?

We can think of the first question as being indicative of a relatively unmethodical exploration of the knowledge-base with the user moving constantly from one item to another. In this mode the user is less concerned with the details of the knowledge available in the system than with attempting to get a broad picture of the items covered. Although a particular item might provide the initial focus, this type of browsing activity is typified by opportunistic choice. The second question reveals the need to do directed investigation for a specific item, or items. In this mode the user has a particular goal and is expecting the tool to be able to provide information abut the breadth and depth of the system's knowledge on the specified item. The third question indicates that the tool be able to support a detailed examination of the knowledge of some specific item.

A useful analogy for thinking about knowledge-base browsing is to consider how one approaches an unfamiliar textbook. Its title tells something about the contents, as does a preface or abstract, but inspection of the contents list gives more detailed information. If one is interested in knowing whether a particular topic is mentioned, then a search of the index will reveal both the number of discussions of the topic and the location of each. Finally, if sufficiently interested, one reads the book to gain an understanding of the material it contains. The goal is to develop a tool that can provide the informational equivalents of the title, abstract, contents list, and index, together with any additional capabilities that are required to give the user the ability to browse the knowledge-bases.

An obvious requirement is that the tool should be able to display frames and evaluation structures in a graphical form, and that the user should be able to control the focus of attention by means of a pointing device such as a mouse. We imagine that the tool interface allows the user to pan both horizontally and vertically, and also to zoom, so providing an extremely flexible mode of interaction with the knowledge. This provides a level of browsing in which the user sees the interconnections of the knowledge in an easily understood form.

Apart from this purely graphical interaction, the browser should allow the user to ask various questions about the connectivity of the evaluation structures, with responses being in the form of mixed graphics and text. To support this type of interaction, the nodes and arcs in the displayed structure will need to be "mouseable." This will allow the user to select a particular element in the network and then ask for

additional information. In the case of a node, which generally corresponds to an attribute, the request could be, for example, to show all the descendants of the node or to show all the ancestors of the node. In response, the tool would display just that part of the structure that satisfies the request. Additional node-related requests could be to display any textual information that relates to a particular node (e.g., the annotations).

In the case where an arc is selected, the tool could allow the user to review the rationale for including the connection between attributes that it represents (i.e., read the text that "explains" the connection), and see the exact form of the connection (i.e., the underlying representation).

Notice that while this type of knowledge interaction goes a long way toward the goal of providing a user with browsing facilities it falls short of the most general form of browsing that we can imagine. Some technique for abstracting the structure and then generating a reduced representation is necessary before we can approach that kind of functionality.

4.3.3. The Analyzer

We believe that a large part of the evaluation process is concerned with trying to understand why the AIS was evaluated the way it was. We call this general activity analysis. To support it, the analyzer needs to have facilities for displaying and interpreting the basic evaluation results, and, perhaps more importantly, facilities for producing various kinds of explanation. Such explanation capabilities should certainly include simple traces of rule invocation, but go beyond them to provide dynamic tracing facilities.

Part of the analysis task involves determining the sensitivity of the evaluation to various assumptions. Performing such an analysis automatically is generally very difficult, so we must rely on a partial, informal, analysis in which the user of the tool controls the direction that the analysis takes. We have identified five separate sensitivity functions that we briefly describe here.

Maximum value achievable by any attribute. This allows the user to see how long chains of propagation will affect the evaluation value assigned to any attribute in the frame.

Maximum contribution from a sub-tree. This is similar to the previous function except that we now focus on the impact from a single chain of argument.

Maximum attribute-to-attribute contribution. This function allows the user to determine the maximum impact any low level attribute can have on a higher level attribute.

Impact of eliminating sub-trees. This function allows the user to examine how the maximum value of an attribute is affected by eliminating certain pieces of the evaluation structure. Using this facility amounts to performing two maximum value computations; one for the original structure and one for the structure without the substructure of interest.

Hypothetical evaluation. This feature allows the user to experiment with the effects of using different measurement procedures or patterns of measurements. We require a

capability which displays all the terminal attributes for a high-level attribute and then allows the user to select the measurement values to be assigned. Once these are assigned, the evaluations can be propagated upwards to the attribute selected.

An equally important set of features required of our analyzer are those that allow the user to examine the dynamic behavior of the evaluation knowledge. The goal here is to provide graphically oriented interaction with the reasoning process itself. So we can imagine a display on which the experimenter sees the evaluation process unfolding and which can be interrupted to allow control parameters to be changed. There could be many variants of this, for example, slow motion tracing of the evaluation structure, snapshots of the tracing at various fixed times, and single step tracing under user control. This type of capability would need to make extensive use of color to show the paths traversed and the evaluation values achieved by any attribute. The experimenter should be provided with a large number of options for controlling the unfolding.

4.3.4. The Evaluator

The evaluator is the tool that actually performs the evaluation. That is, it takes measurements and interprets them in the context of the evaluation structure. We think of it as being normally invoked from within the other tools. So, for example, within the editor local evaluations may be performed to perform various kinds of syntax checking during the modification of knowledge. Similarly, many of the sensitivity functions available within the analyzer are examples of partial evaluations. These tasks require the evaluator to work just with pieces of the overall evaluation context rather than its whole, but, of course, the evaluator can be invoked directly when an actual evaluation is to be performed.

The evaluator is thus the equivalent of the inference engine in a conventional AIS. More accurately, it is equivalent to a collection of engines since we allow multiple interpretation methods both within frames and across frames.

4.4. DESIGN AND IMPLEMENTATION

The object-oriented paradigm appears to be a natural implementation for the system we have described above. All of the entities we have defined in our environment can be thought of as objects, and the operations we wish to perform can be modelled using message passing. So for example, our concept of an attribute can be represented by defining a generic attribute object that has properties which are permanent and context independent (i.e., part of the definition of the attribute), as well as properties that are only recognized when the attribute is instantiated as part of an evaluation frame.

In Figure 4-2 we show a generic attribute object and and in Figures 4-4 and 4-5 shows its instantiations in two different evaluation frames.

For our example, we have defined a generic attribute called *Vendor-support*. A generic attribute is a "definition," specifying the properties which are independent of any evaluation frame in which the attribute might be used. We deliberately chose a

Slot Name	Slot Value
Name:	Vendor-support
Depends-on:	(Handholding, Expertise, Timeliness)
Annotations:	General description of what vendor support features are important to us. Might consider adding a sub-attribute for "Ease of Access" to the support staff. ...

Figure 4-2 Generic Attribute Object

Slot Name	Slot Value
Name:	Vendor-support
Value:	
Value-type:	Linguistic-var
Output-mapping:	
Depends-on:	(Handholding, Timeliness)
Combination-fcn:	Lv-combination-fcn-1
Annotations:	Modified from general description of vendor support. This version deletes the expertise dependency because novice users are usually more concerned with breadth of support.

Figure 4-3 Novice User Instantiation

Slot Name	Slot Value
Name:	Vendor-support
Value:	
Value-type:	Multi-attr-utility
Output-mapping:	
Depends-on:	(Expertise, Timeliness)
Combination-fcn:	My-mau-combination-fcn
Annotations:	Created 5/15/87 by J. Smith. Tailored to experienced users. Definitely consider adding dependency on "Ease of Access" to support staff.

Figure 4-4 Expert User Instantiation

definition that is independent of any system which might be an evaluation target. It is this generic definition that would be stored in one of our evaluation system's attribute libraries. The context-independent properties of an attribute are *Name*, *Depends-on*, and *Annotations*. *Name* is simply a label by which the attribute is referenced. *Depends-on* gives the names of other attributes which are important components of this one. Hence, in the example, the attribute's author has implied that "in order to evaluate *Vendor-support* we must know something about the 'Expertise' of the support staff, the 'Handholding' services provided, and the 'Timeliness' of response to customer's questions."

When the generic attribute is instantiated (by having its definition inserted into an evaluation frame) we find that there are several additional properties which have meaning in the context of the parent frame. Now there are concepts of a *Value* for the attribute, the *Value-type*, *Output-mapping* , and a *Combination-function*. *Value* is the

result of evaluating this particular attribute, and therefore does not exist until the evaluation is executed or a "result" is entered directly by the user. *Value-type* indicates how *Value* will be represented. Some choices might be linguistic variables, points on a discrete 5-point scale, etc. *Output-mapping* is a function for converting between value types. That is, when the user has specified (via the *Value-type* field) that *Value* should have a representation which happens to differ from the representation used by the attribute's measurement procedure. *Combination-function* is the function used to combine inputs into a single value. An attribute's inputs are either the *Value* fields of the attributes in its *Depends-on* list, as is the case in our example, or else there is a single input which is generated by a *Measurement-procedure*. The second case applies to terminal attributes, those attributes which have an empty *Depends-on* list. For terminal attributes, *Measurement-procedure* replaces *Combination-function*.

The context-dependent fields, *Value-type*, *Output-mappings*, and *Combination-function/Measurement-procedure*, are inherited from higher order attributes and from the evaluation frame itself according to preferences specified by the user. When a user feels that an attribute definition is accurate enough to be considered a model for the attribute it represents, or if in the case of a terminal attribute there is a standard measurement procedure for the attribute, the *Combination-function* (or *Measurement-procedure*) field may be treated as a permanent part of the definition.

The two instantiated attributes shown in Figures 3-4 and 3-5 are from frames called "Novice User" and "Expert User". The differences are relatively straightforward; the Novice User has chosen to represent the attribute value using linguistic variables, while the Expert User opted to use Multi-Attribute Utility. Consequently, the Novice User's combination function is "Lv-combination-fcn-1", presumably chosen by the system as a default given the choice of representation. The Expert User has decided not to use a default combination function and instead is providing one of her own called "My-mau-combination-fcn". Both users have actually gone a bit farther and edited the instantiated attribute's *Depends-on* field to reflect their own view of what *Vendor-support* depends on. The novice user here is concerned far more with handholding and timely response to questions than to deep expertise on the part of the support staff, while the expert user feels she is well past the handholding stage. They have also edited the *Annotations* field, making note of the reasons for their preferences and modifications in constructing this version of the *Vendor-support* attribute.

Either of the different versions of *Vendor-support* in the example could be saved and copied into an appropriate attribute library. Saving under different names would yield alternative versions of the attribute, while saving under the name *Vendor-support* would replace, or redefine, *Vendor-support*.

Finally then, the implementation of an environment such as the one we have described does not appear to pose any particular problems. Indeed, given the requirements defined by our methodological approach and the top-level design just described, we believe that either a commercially available expert system building tool,

such as KEE, or a lower-level object-oriented language like Commonloops or SOPE, would be an appropriate base on which to construct our environment.

Chapter 5

ASSESSMENT OF LIKELIHOOD OF SUCCESS

As a result of developing the examples and the prototype environment design described in the earlier chapters, we conclude that our proposed approach to the performance evaluation of AISs has a high likelihood of achieving useful results.

The methodology we have proposed does not depend upon the development of new evaluation techniques but, rather, relies upon the structuring of knowledge about performance evaluation. Since we have considerable experience with the knowledge acquisition and representation task, and since our examples show that this domain is actually rather straightforward, we can assume that the development of useful knowledge structures will not pose any significant problems.

Similarly, the design for the evaluation environment draws upon well known implementation techniques and, if based upon a commercially available ESBT or object-oriented language, we expect that the construction of a workable prototype system will carry a very low risk.

Of course, the final test of the utility of our approach can only be assessed in the light of serious attempts to evaluate some operational AISs. We believe that exploration of any of the tools and systems currently being sponsored by DARPA (e.g., KEE 3.0, ABE, ALV, ALBM) would be appropriate for pilot experiments. Careful monitoring of the use of the environment during these evaluations would enable further refinements and additions to be made to our methodology. In the long term, we would expect this research and development to lead to a more detailed understanding of the process of evaluation for AISs in general, and to generate specific recommendations for evaluation procedures for certain classes of AIS.

Chapter 6

SUMMARY AND RECOMMENDATIONS

In this final chapter, we summarize the work performed during the current effort and make some recommendations for Phase II research and development.

6.1. PROJECT SUMMARY

6.1.1. A Methodology for Evaluating AISs

A major part of our effort during this project has been to develop a methodology for evaluating artificial intelligence systems. We started with the premise that it is usually inappropriate to ask the question, Is the AIS intelligent? Much more interesting, and useful, are questions such as, Does the AIS help the user?, In what ways does it help?, and, How can we quantify these benefits? Our goal, then, has been to develop a methodology that focusses on the issue of what constitutes an appropriate set of evaluation criteria, and describe a technique for organizing and manipulating such information. Our view is that the proper structuring of performance evaluation knowledge is the critical step in performing an evaluation. The result is an approach that introduces the idea of an evaluation frame as the central element in the knowledge representation. The frame sets the context for the evaluation and is intended to reflect the many dimensions of the evaluation process; the nature of the evaluation question being asked, the characteristics of the evaluator, the type of AIS being evaluated, and the stage of development of the AIS.

6.1.2. Example Evaluations

To test the validity of our approach, we developed a number of example evaluations. Our goal was to both to explore the consequences of our concept of an evaluation frame and to develop some preliminary needs and requirements for a prototype evaluation environment. We considered three basic examples. In the first, we explored the problem of selecting an expert system building tool for two different applications. In the second, we developed an evaluation for a science and technology analyst's assistant with respect to three different questions posed by the system developers. And in the third, we constructed an evaluation of an advanced full-text information retrieval system from the perspective of the end-user and the system developer.

6.1.3. A Prototype Environment

Using the insights gained from the example evaluations, we were able to develop a preliminary system concept for a prototype evaluation environment. This may be characterized as an "Evaluator's Workbench." It contains a number of tools for editing evaluation structures, browsing libraries of evaluation knowledge, and performing and analyzing evaluations. The style of interaction is highly interactive and graphically oriented. The design does not require the development of any significantly new system implementation techniques and so can be implemented using an object-

oriented language, or perhaps a commercially available expert system building tool such as KEE.

6.1.4. Achievements

The main achievement of this effort is the recognition that evaluation of AISs is, in principle, no different from the evaluation of any other computer-based tool for decision-making and information analysis. What is important, however, is that the we recognize that evaluations are performed under different assumptions, and that making these assumptions explicit is extremely valuable is using the results of the evaluation. Accordingly, our methodology is explicitly designed to support the structured organization of assumptions and performance criteria.

6.2. FUTURE RESEARCH AND DEVELOPMENT

6.2.1. System Design and Implementation

The first recommendation, naturally, is that we proceed with the design and implementation of the evaluation environment described in Chapter 4. We would expect that a preliminary version of the environment could be developed within a period of twelve months. The main effort would be in the design of the user interface and the provision of initial versions of the tools.

6.2.2. Pilot Experiments

The second recommendation is that a number of pilot experiments be performed using the environment. These will serve both to provide further validations of the methodology and to refine the design of the environment itself. An additional likely benefit of performing these experiments would be the development of the beginnings of standard evaluation procedures for AISs. Possible candidate tools and AISs sponsored by DARPA with which ADS has some experience are KEE, ABE, ALV and ALBM. Some, or all, of these could be the focus of experiments designed to develop and test attribute spaces, evaluation frames and interpretation methods.

6.2.3. Evaluation Process Research

Finally, we recommend that additional research and development be undertaken on the process of evaluation. We have identified three basic research tasks that would assist in our understanding of how to perform the performance evaluation of AISs:

(1). We need to develop definitions and measurement procedures for evaluation attributes that are specific to AISs. In Chapter 2 we alluded to attributes of knowledge (e.g., expressiveness, consistency, completeness) and attributes of inference (e.g., soundness, support of multiple reasoning paradigms) that seem to be properties of AISs alone. Measurement of these qualities is extremely difficult, not least because we do not have clear definitions for them. Research into appropriate metrics, perhaps based on work on the evaluation of human decision makers, would be extremely beneficial.

(2). A more technical line of research is to specify formal procedures for combining evaluations generated using different interpretation methods. Our methodology supports such a concept, but the we have few theoretical results that tell us how to take the output from a MAUT evaluation, say, and combine it with a subjective evaluation that uses a seven point scale.

(3). Finally, we might perform research on more detailed models of the evaluation process itself. In Chapter 1 we talked about evaluations informing decisions. What is required are more detailed cognitive models of the use of evaluations. In particular, we need to understand how issues such as the importance of the decision are reflected in the evaluation structures. Such models would help us develop an evaluation environment that is more responsive to the needs of evaluators. For example, we can imagine that an advanced version of the environment could make use of these cognitive models to actively make suggestions about the form and content of the evaluation frames.

REFERENCES

Abram, J.M., F.X. Lanzinger, J.R. Payne, R.M. Tong, R.P. Wishner. (1983) *A Methodology for Determining Fusion System Architectures.* Rome Air Development Center, Final Technical Report RADC-TR-83-6.

Bonissone, P.P. (1978) *A Pattern Recognition Approach to the Problem of Linguistic Approximation in Systems Analysis.* Electronics Research Lab., Univ. California at Berkeley, Technical Memo. UCB/ERL-M78/57.

Chong, C-Y., S. Mori, T. Miltonberger, D.S. Spain, R.M. Tong. (1984) *Fusion System Architecture Analysis and Synthesis.* AI&DS Final Technical Report TR-3021-01.

Cromarty, A.S. (1985) What are Current Expert Systems Tools Missing? *Proc. 13th IEEE Computer Society Int. Conf. (COMPCON85)*, San Francisco, February.

Cromarty, A.S., T.L. Adams, G.A. Wilson, J.F. Cunningham, C.J. Tollander, M.R. Grinberg. (1986) Distributed Database Considerations in an Expert System for Radar Analysis. In: L. Kerschberg (ed.), *Expert Database Systems.* Benjamin/Cummings Publishing Co.

Dubois, D., H. Prade. (1980) *Fuzzy Sets and Systems: Theory and Applications.* Academic Press.

Efstathiou,. J., R.M. Tong. (1982) Ranking Fuzzy Sets: A Decision Theoretic Approach. *IEEE Trans. Systems, Man, and Cybernetics.* SMC-12:655-659.

Forman, E.H., T. Nagy. (1985) A Multicriteria Model to Select an Expert System Generator. *Proc. IEEE Symp. on Expert Systems in Government.* McLean, VA.

Fung, R.M., G. Stachnic, T. Bache. (1987) *Inference Strategy Development for an Intelligent Seismic Monitoring System.* Advanced Decision Systems, Technical Memo, TM-1163-01.

Gabriel, R.P. (1985) *Performance and Evaluation of Lisp Systems.* MIT Press, Cambridge, Mass.

Gaschnig, J., P. Klahr, H. Pople, E.H. Shortliffe, A. Terry. (1983) Evaluation of Expert Systems: Issues and Case Studies. In: *Building Expert Systems.* F. Hayes-Roth, D.A. Waterman, D.B. Lenat (eds.) Addison-Wesley Publishing Co., Reading, Mass.

Gevarter, W.B. (1987) The Nature and Evaluation of Commercial Expert System Building Tools. *Computer.* 20(5):24-42.

Karna, A., A. Karna. (1985) Evaluating the Existing Tools for Developing Expert Systems in the PC Environment. *Proc. IEEE Symp. on Expert Systems in Government.* McLean, VA.

Keeney, R.L., H. Raiffa. (1976) *Decision with Multiple Objectives.* Wiley.

McCune, B.P. (1984) Future Requirements for Tools for Building Knowledge-Based Systems. *Proc. IEEE Computer Soc. 8th Int. Computer Software & Applications Conf.* Chicago, Illinois.

Saaty, T.L. (1980) *The Analytic Hierarchy Process.* McGraw-Hill.

Tong, R.M., P.P. Bonissone. (1980) A Linguistic Approach to Decision Making with Fuzzy Sets. *IEEE Trans. Systems, Man, and Cybernetics.* SMC-10:716-723.

Tong, R.M., V.N. Askman, J.F. Cunningham, C.J. Tollander. (1985) RUBRIC: An Environment for Full Text Information Retrieval. *Proc. 8th Int. ACM SIGIR Conf. on R&D in Information Retrieval,* Montreal, Canada.

Veit, C.T., M. Callero. (1981) *Subjective Transfer Function Approach to Complex System Analysis.* Rand Corp., Report R-2719-AF.

Wall R.S., A.W. Apon, J. Beal, M.T. Gately, L.G. Oren. (1985) *An Evaluation of Commercial Expert System Building Tools.* Texas Instruments, Computer Science Laboratory Technical Report 85-30, Dallas, November 1985.

Waterman, D.A., F. Hayes-Roth. (1982) *An Investigation of Tools for Building Expert Systems.* Rand Corporation Report R-2818-NSF, Santa Monica, June.

Zadeh, L.A. (1965) Fuzzy Sets. *Information and Control.* 8:338-353.

Zadeh, L.A. (1973) Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. *IEEE Trans. Systems, Man, and Cybernetics.* SMC-3:28-44.

Zadeh, L.A. (1975) The Concept of a Linguistic Variable and its Application to Approximate Reasoning. *Information Sciences.* 8:199-249.

APPENDIX A: ASTA OVERVIEW

This appendix contains a paper on the ASTA system presented at the *1st Int. Conf. on Expert Database Systems.* Kiawah Island, South Carolina, October 1984, and reprinted in L. Kerschberg (ed.). *Expert Database Systems*, Benjamin/Cummings Publishing Co., 1986.

DISTRIBUTED DATABASE CONSIDERATIONS IN AN EXPERT SYSTEM FOR RADAR ANALYSIS

Andrew S. Cromarty, Thomas L. Adams, Gerald A. Wilson, James F. Cunningham, Carl J. Tollander, and Milton R. Grinberg*

Advanced Decision Systems, 201 San Antonio Circle, Suite 286, Mountain View, California 94040, USA

* Heuristic Programming Project, Department of Computer Science, Stanford University, Stanford, California 94305

ABSTRACT

A knowledge-based research prototype has been created for analyzing existing radar systems. The expert system, called ASTA, interactively accepts values for system attributes, subsystem attributes, or signal parameters, and then incrementally infers the value of as many other attributes as it can. The ASTA knowledge base represents the first known effort to structure information about radar design as a function of observable operating characteristics.

We present the architecture of ASTA as a working example of an expert system that faces and solves several practical problems in the marriage of an expert system to distributed databases. ASTA ensures database concurrency across differently structured databases, and yet meets the pragmatic constraints of integrity and timeliness of the data required by the expert system. This is achieved in part by the use of a distributed message-passing architecture with explicit time-stamping of messages. We also discuss ASTA's current interface to external DBMS's and propose a new technique which would enable the inference engine and the DBMS to cooperate more closely in the selection of data when the query is imprecise.

INTRODUCTION

The challenge of developing consultation systems which provide knowledge-based support for people performing complex tasks is both formidable and important. Real problems require an integrated use of problem-specific knowledge, current and background data, and effective tools for handling the cooperation between the human user and the computer system. In this paper we describe the knowledge representation and data base issues, and their solutions, which have been addressed in an ongoing project. The objective of this project is to develop a knowledge-based consultation system, called the Assistant for Science and Technology Analysis (ASTA), which assists individuals attempting to predict the architecture and performance characteristics of modern electronic systems given only a description of the observed operating behavior of the system.

Expert Database Systems, Larry Kerschberg, Editor

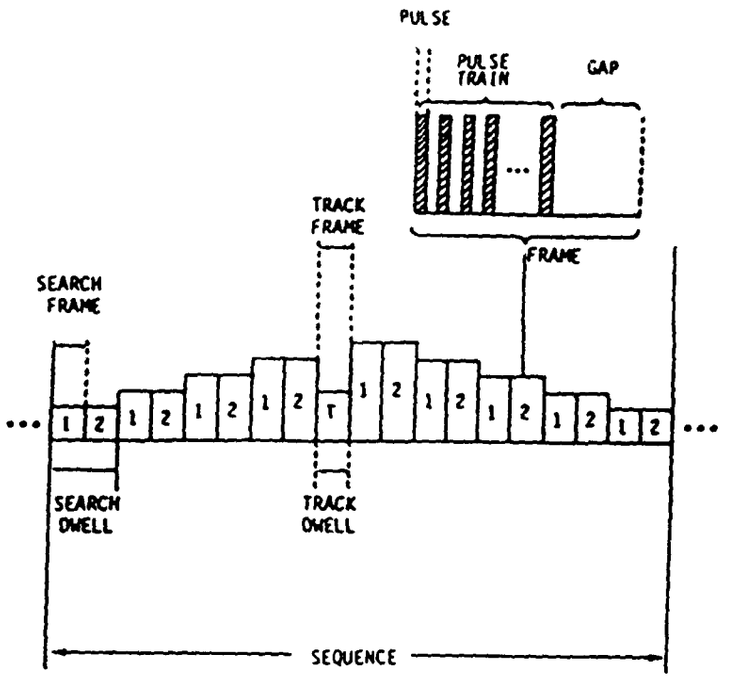
Three aspects of this project are discussed in this paper. First, we describe the approach employed in representing the knowledge and data required for the inference engine. The data, as well as some of the knowledge, must be maintained in a form appropriate for interaction with the human user through our MPS [1] interface. At the same time, these data must be available in a form appropriate for the inference engine to employ for hypothesis generation. Second, because the same data must be maintained in each of the two major processes (the interface and the inference engine) of the system, there must be a means to maintain the consistency of the two representations. Third, the inference process also requires access to supporting data maintained by one or more DBMS's external to ASTA. Each of these aspects is discussed below.

KNOWLEDGE-BASED RADAR SYSTEM ANALYSIS: A BRIEF OVERVIEW

ASTA is tailored to radar systems by virtue of its database of symbolic facts and heuristics that define how new information should be inferred from existing evidence and previously reached conclusions. Because of the heuristic nature of much of the information that such a database contains, it is properly called a ***knowledge base***. The ASTA knowledge base represents the first known effort to structure information about radar design as a function of observable operating characteristics, rather than from the point of view of the design process. The knowledge comes from expert radar designers and analysts and from radar design handbooks. This includes general knowledge about radar systems, such as the physics of radar signals and the relationship between different components of radar systems. ASTA also has knowledge about itself: it contains explicit rules identifying (a) the ways it can use logic to solve problems, (b) the problems that are worth solving, and (c) the order in which interesting problems can most effectively be solved.

ASTA's knowledge may express either numeric or logical relationships. By separating declarative knowledge about radar physics and radar analysis problem-solving techniques from the generic inference-related control aspects of the computer program (its *inference engine*) that operates on that knowledge, ASTA facilitates inspection of its knowledge base and its line of reasoning in order to explain why new values were inferred. Further, its body of radar facts and analysis techniques can be modified by non-programmers, and it can operate robustly in the face of the partial or errorful data that typify analysis tasks.

Radar is an echo-location system wherein pulses sent out at a given frequency (the Pulse Repetition Frequency) are listened for as they return after reflection off environmental objects. The presence of an object is indicated by a strong return, and its velocity and range can be determined by the length of time from transmission until the echo is heard and by the way the object alters characteristics of the transmitted radar signal. A pictorial representation of a typical radar waveform and scanning sequence, which constitutes the raw data described in the formatted reports that ASTA accepts as



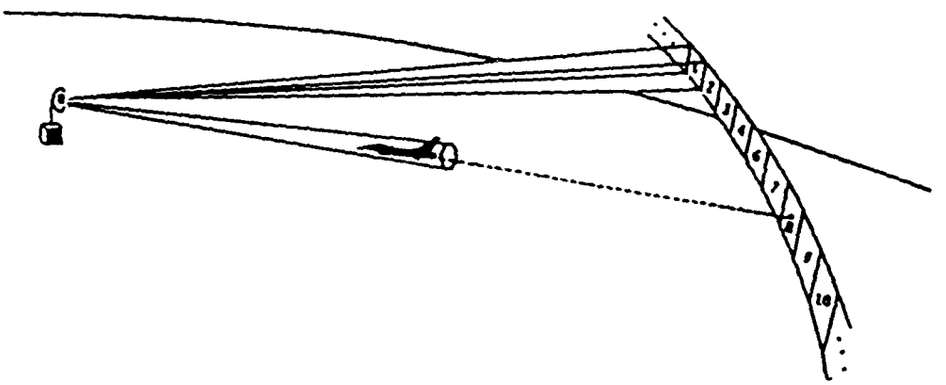


Figure 1: A Pictorial Representation of Waveform and Scan Pattern

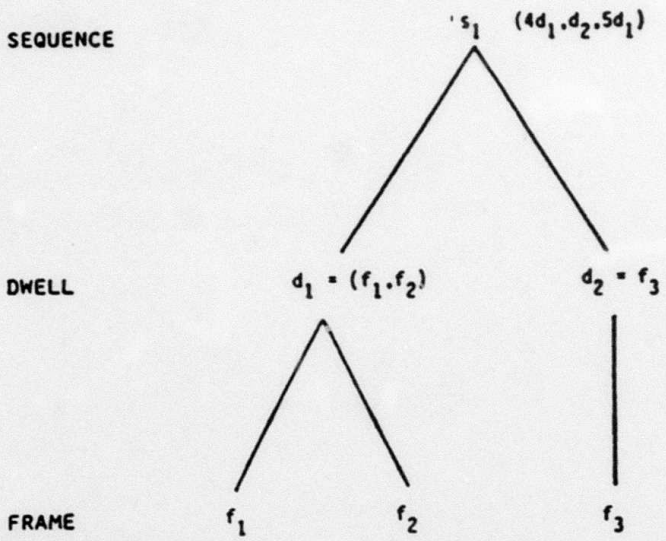


Figure 2: ASTA's Internal Representation of Waveform and Scan Pattern

input, is given in Figure 1. (A tree diagram indicating the corresponding internal structures we use to represent such data is shown in Figure 2.) ASTA represents the waveform and operational characteristics according to the following taxonomy:

- *Frame level information*: A description of the basic waveform parameters of each of the pulse trains used in the system.

- *Dwell level information*: A description of how the individual pulse trains are used together to achieve the intended radar function. In the example, two frames are used sequentially to search for as yet unseen objects and a single frame is used to track an object already observed. During a dwell, the radar antenna is focused on (dwells on) one location in the sky and is attempting either to search that location

for undetected objects or to track an already-observed object believed to be at that approximate location.

- *Sequence level information*: A description of the method by which the operational modes of the radar are multiplexed, i.e. the way that search and track dwells are interleaved to allow the radar system to both keep track of objects it does know about and find new ones it has not yet detected. In the example, only one object is being tracked, and the radar search mode is interrupted periodically every tenth spatial dwell to accomplish tracking. (This is an important problem for airports where a single air traffic control radar must become aware of new planes that are coming in for landing while known planes are being observed, for example, when multiple planes may be attempting to land at once.)

An example of the rule knowledge contained in ASTA's knowledge base is given in Figure 3, which portrays a fragment of the rule graph structure that ASTA employs to infer the kind of radio tube used in the amplifier of the radar transmitter under study, given only a description of the characteristics of a signal observed to have been emitted by the radar. Figure 4 provides the corresponding modified predicate calculus form that MRS uses to represent the knowledge depicted in Figure 3, with their English equivalents. The lower-level frame parameter processing rules used to establish the maximum instantaneous bandwidth and the nominal frequency are not shown for the sake of brevity. Rule 1, which defines the fractional instantaneous bandwidth, is enabled whenever the instantaneous bandwidth and the nominal frequency have been previously calculated and stored in the collection of facts describing the current state of knowledge of the system. Thus the rule contains not only the expression used for calculating the result, but also the more general information that the expression can only be solved for the independent variable, if the two dependent variables are known. Rule 2 states that, if the number of elements in the set of signal frames satisfying the high duty cycle and high PRF condition is equal to the total number of signal frames (thereby establishing that every signal frame satisfies the condition), then the radar system employs pulsed Doppler modulation. This example shows that (1) the conditions for triggering a symbolic conclusion can depend on an arithmetically calculated precondition and (2) the set of objects over which the constraint is satisfied can be specified in the rule. Rule 3 is of the purely symbolic type commonly employed in expert systems. Rule 4 illustrates how the satisfaction of a symbolic constraint and an arithmetic constraint can be used as preconditions for a symbolic conclusion.

This example illustrates how ASTA can employ a wide variety of mixtures of symbolic and arithmetic expressions to obtain its conclusions, and also how two parallel reasoning paths (pulse characteristics to infer bandwidth and PRF to infer pulsed Doppler) can be combined to produce the desired conclusion (klystron tube type). Readers interested in the radar analysis problem will find more detailed explanations and examples in [2].

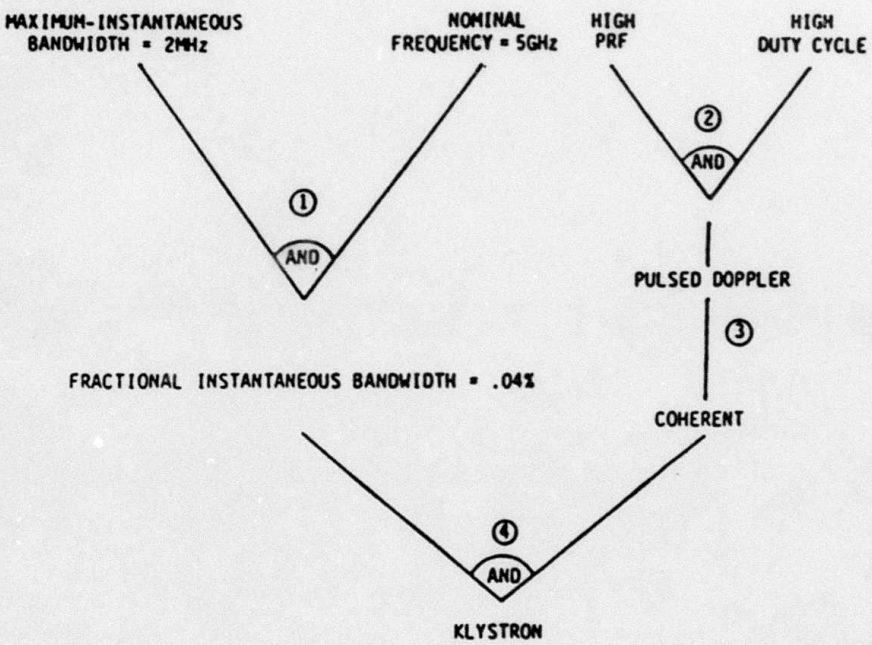


Figure 3: Rule Graph Fragment for Transmitter Output Tube Type Determination

(1)
```
(if (and (inst-bandwidth radar-xmtr $x)
         (nominal-frequency radar-xmtr $y)
         (is (// (// $x $y) 1000.0) $z))
    (frac-inst-bandwidth radar-xmtr $z))
```

"The fractional instantaneous bandwidth is the instantaneous bandwidth divided by the nominal frequency of the transmitter."

(2)
```
(if (and (setof $n (and (radar-signal-frame $n)
                        (PRF $n $a)
                        (> $a 8)
                        (duty-cycle $n $b)
                        (> $b 3))
                $q)
         (setof $m (radar-signal-frame $m) $p)
         (length $p $c)
         (length $q $d)
         (= $c $d))
    (modulation-type radar-system p-dop))
```

"If every frame observed has a PRF greater than 8 KHz and a duty cycle greater than 3%, then the radar system employs pulsed-Doppler modulation."

(3)
```
(if (modulation-type radar-system p-dop)
    (coherent radar-xmtr))
```

"Pulsed-Doppler systems normally employ coherent modulation."

(4)
```
(if (and (frac-inst-bandwidth radar-xmtr $x)
         (< $x .05)
         (coherent radar-xmtr))
    (tube-type radar-xmtr klystr ))
```

"The klystron is the preferred transmitter tube type for a narrowband, coherent system."

Figure 4: Inference Rules for Transmitter Output Tube Type Determination

AN OVERVIEW OF THE ASTA IMPLEMENTATION

The ASTA implementation comprises three parts: the interactive, menu-driven, forms-oriented user interface based upon the Multipurpose Presentation System (MPS) [1]; the Metalevel Representation System (MRS) [3] for constructing rule-based systems; and the COP control and communications system [4] to perform message routing and system task planning functions. The underlying architecture for ASTA is thus a functionally partitioned, multiple-process message passing design, depicted in Figure 5. The separation of the expert system *per se* from its user interface was motivated by several goals:

- The desire for a *clean modular design* with effective information hiding. The conviction that a module should "do only one thing and do it well" argues for making accessible to a given module exactly and only those data it needs to perform its job, without providing access to ancillary data and hence introducing a risk of inadvertent data collisions or unintended data dependencies.

- The need for a *reliable incremental development environment* with minimal subpart interaction. An important aspect of the ASTA specification was the requirement that the knowledge base be amenable to continued development well after the user interface had stabilized; this can be achieved by divorcing the rule set from the data base that supports user interface activity and defining a narrow bandwidth communication mechanism between the two.

- The desire to employ *locally optimal representations for each subtask* of the problem that ASTA is trying to solve. That is, separation of the component modules allows different data representation techniques to be employed rather than forcing reliance upon a single representation that optimally supports only one (or neither) of the two tasks, reasoning and user interaction, that ASTA must perform.

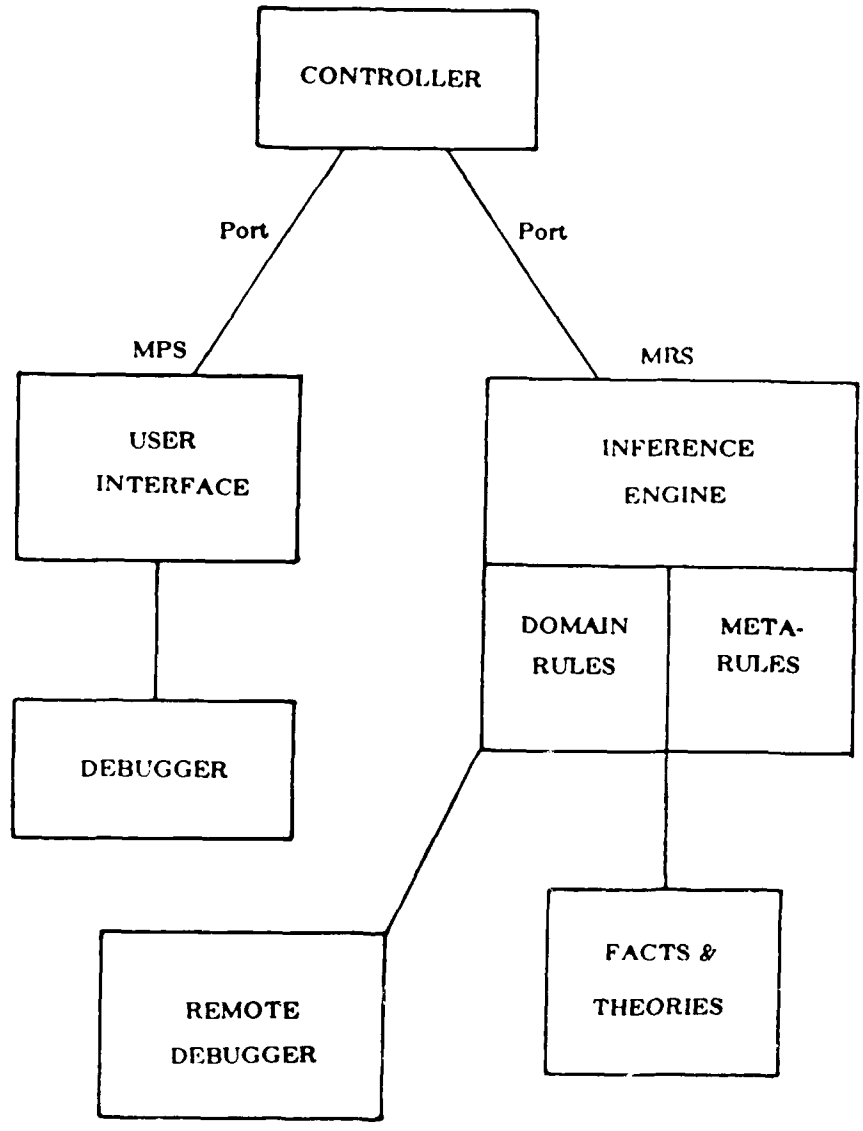


Figure 5: The ASTA Multiple-process Message Passing Architecture

THE EXPERT KNOWLEDGE DATABASE

The knowledge base of facts, rules, and metarules is contained entirely within a process that is assigned the responsibility of pursuing the inferences required to support the analysis. MRS provides the general purpose inference engine and data base capability used within ASTA to store and derive radar system parameters. It supports a rule-based approach that employs a knowledge base of facts and rules along with a flexible control structure used to guide the inference strategy. MRS is a domain-independent reasoning and representation system in which knowledge about any field may be represented. In the ASTA system, MRS stores and maintains all of the domain specific knowledge of radar systems including the initial default values for physical constants, the current known radar parameters that have been entered by the analyst, the radar parameters that have been derived from one or more known parameters, the rules used to relate the radar parameters to each other, and the meta-level knowledge used to control the use of the rules and data. The information that the system uses may be numeric or symbolic in nature. MRS has the ability to make mathematical calculations or to draw inferences based on symbolic information to derive new symbolic information. Furthermore, symbolic information can be used to select the appropriate form of a calculation or to provide constraints on the range of values in the terms of an equation. Because the analysis domain is characterized by information that is informal, imprecise and incomplete, we stores all of this knowledge declaratively, rather than procedurally. By storing the inferencing procedures declaratively, the system has the ability to reason with them, manipulate them, and use them only when enough information exists to derive new data from the existing data. The inference process is therefore very flexible and can use the information it has available to make all the conclusions it can, but will not be hindered or rendered useless when certain radar parameters are unknown.

The **stash** operation is used to store *assertions* (facts and rules describing the current problem domain) in the database. These assertions are stored as n-tuples in MRS. Intermediate inferences need not always be stored in the database, so the meta-level control is used to define when and where assertions are stashed. For the ASTA application, we have chosen to represent the radar system parameters in 3-tuples consisting of a property, object, and value, where the object can be thought of as an index into a table of values for all objects with the specificied property. For example, the 3-tuple (prf f1 15) states that the property "prf" has the value "15" for the radar frame indexed by "f1". Assertions are retrieved from the database by the "truep" ("truth" predicate) operation. It queries the database for evidence concerning the validity of a statement given the current context. The system will determine if this statement is true by searching the database for it; if the fact is not present, MRS will then try to infer the validity of the statement from the database using the known facts along with the rules of inference.

The rules in MRS are of the common *if-then* production rule form, with the *if* clause consisting of one or more antecedents (preconditions) that, when true, imply that

the consequent statement associated with the *then* clause is true. Any logical proposition can be encoded as a set of rules in this form by first putting it into conjunctive normal form and converting each disjunct into an appropriate rule. MRS then allows these rules to be used in two ways: either in a data-driven, forward chaining direction, from antecedents to consequents, or in a goal-directed, backward chaining direction, from consequents to antecedents. In the data-driven direction, the system will try to match the information it has against the antecedents of the rules, and when successful, will add the consequent of the matched rules to the database. Conversely, when performing goal-directed reasoning, the system will hypothesize that a particular goal is true and try to find rules that contain the desired goal in its consequent. If such rules are found, the system will attempt to match the antecedents of these rules against the database, and if at least one rule is successful, will then add the desired consequent to the database. If no such rule succeeds, the system may try to find further rules with the unsuccessfully matched antecedents in the consequent of another rule and attempt to determine the truth of these using the same procedure. For example, ASTA has the following rule in its database:

```
(if (and (pulse-modulation $f psk)
         (chip-duration $f $p))
    (compressed-pulse-duration $f $p))
```

The antecedent assertions are the clauses with the properties "pulse-modulation" and "chip-duration", while the consequent clause has the property "compressed-pulse-duration". (Variables are distiguished by a "$" preceeding the name, and can match against any instantiated term in another clause.) In this example, the first antecedent has the variable "$f" which will match the first index it finds with the property "pulse-modulation" that has the value of "psk". When using this rule in the forward chaining direction, the system will try to match these antecedents, which must both be true since they are joined by the "and" operator, against the database. If successful, it will assert the consequent with the property "compressed-pulse-duration" using the same instantiations for the variables that it used for the antecedents. As an example of applying the previous rule in the backward chaining direction, suppose it is desired to determine the compressed-pulse-duration of a particular frame "f1" as the instantiation for "$f". The compressed-pulse-duration can be thought of as a goal that the system will try to prove using the database of current assertions along with the applicable rules. MRS does this by instantiating the antecedents of this rule; if these can be matched against the database, MRS will use the same instantiation in the consequent clause to assert the fact. If one or more of the antecedents does not occur in the database, the system may post them as new goals and try to prove them true using the same backward chaining mechanism.

The method of search and chaining direction is selected by the ASTA design team on a case-by-case basis and implemented through the use of the meta-level rules (with default search techniques employed where appropriate). Meta-level control operations provide control over the use of the rules as well as a means of manipulation of the

environment of assertions that are currently valid. The meta-level consists of both assertions and rules of the same form as the base level rules and assertions, stored in the same database. The meta-level assertions dictate how the system uses rules. For example, they can proscribe the use of backward chaining or forward chaining mechanisms, control the stashing of results in the database, or specify whether it is appropriate to seek out more than one instantiation of a particular goal.

Meta-level assertions can be used to affect the current context of applicable databases that are valid. MRS allows multiple databases, called "theories", to be used for stashing both rules and assertions. The meta-level defines which theories are currently "active" (searchable) and how to change the state of a theory between active and inactive. The meta-level rules may contain context dependent conditions that determine when the meta-level assertions are applied. These rules have the same form as the base level rules; however, the base level rules embody knowledge about the particular domain, whereas the meta-rules adapt the use of those rules to the current situation. This mechanism allows the system to understand its effect in the current context and then adapt to the constantly changing situations that it encounters.

These database mechanisms allows ASTA to work with multiple, mutually exclusive hypothesis simultaneously. Competing hypothesis data are manipulated in different databases, and storage and retrieval are managed using meta-level rules. Specifically, the ASTA system will activate a theory to store the hypothesized values and values derived from inferences made using these hypothesis. The theory may also contain specific rules that are only activated with the theory and, therefore, only with the one set of data associated with the particular hypothesis. If the hypothesis is found to be valid, the assertions made in its local theory may be moved to a more global theory, or this theory may simply be moved into a new context within the global framework. If the hypothesis is found to be invalid, the assertions may be discarded or moved to an inactive state which will not affect the other theories.

The ASTA knowledge base also contains a network of *justifications*, used to generate explanations. Every assertion in the system has an associated explanation of how it was derived, either from the user as an input parameter, from a system constant, or a value derived from a rule. For each assertion that is derived from a rule, the explanation database must save the rule that was used to derive the fact, the current meta-level control that was in effect when the rule was fired, and a binding list of the instantiations of the variables in the rule. From this information, the system can trace the derivation of a value, and the rules that were applied to the input parameters to infer the value that is in question.

THE USER INTERFACE DATABASE

The MPS user interface system maintains a database, physically but not logically independent of the MRS knowledge base, that is focused on the specific problem of providing efficient access to the data that specify the state of the interaction with the user.

The MPS user interface is object oriented. Text input boxes, captions, and menu selections are stored in the MPS data base as dynamic record structures. These objects are the components of a special kind of object called a *presentation surface*, which handles their display and manipulation by the inference engine and the user. MPS is composed of a number of presentation surfaces along with a toolkit of primitives for their creation, modification, and manipulation.

At the present time, the allowable object types are:

- **caption**: simple text strings placed the screen at a specified location.
- **selection**: like captions, but the user may position the cursor on the text string and select it as a menu option.
- **input**: a reverse video box placed on the screen at a specified location, into which the user may type text strings.

A presentation surface, and the form or menu that it displays, are created by the function **NewForm**, which sequentially evaluates the object specifications for captions, selections and inputs found in a template file. These specifications take the form of the relations:

(**caption** *coordinates unique-id initialization-string-value*)

(**selection** *coordinates unique-id initialization-string-value*)

(**input** *coordinates unique-id field-size field-type*)

The *field-type* argument permits some simple type checking for integer, floating point and string values. The **caption**, **selection**, and **input** abstract relations are grouped together to form presentation surfaces using the relation

presentation-surface-id = (*bptr clist slist ilist*)

where *bptr* is a pointer to the bitmap of the display form, *clist* is a list of unique caption identifiers, *slist* is a list of unique selection identifiers, and *ilist* is a list of unique input

identifiers. Each such identifier is in turn a relation:

identifier = (*row column value* [*field-size*] [*field-type*])

We thus achieve *object orientation* in part through the mechanism of unique naming, making each data object a uniquely identifiable object with an existence of its own, i.e. not dependent on its relationship to other data objects for its identity. Some of the advantages of this approach over the use of primary keys as a data object unique identifier are discussed by Codd [5], where he refers to them as *surrogates* corresponding to an implicit *E-attribute* (entity-designation attribute) of the data object (tuple) of interest.

Each radar signal component structure (e.g. frames, dwells, sequences) and each radar system component structure (e.g. antenna, transmitter, receiver) is mapped by **NewForm** onto a presentation surface at initialization time. There is thus a one-to-one correspondence between the "f1" radar signal frame data object of the inference process and the form displayed by the presentation surface called "f1". The properties of the components are mapped onto an MPS "input" record.

The inference process may manipulate MPS structures through a variety of primitives, although most operations of interest can be achieved using the primitive relations **Show**, **SSet** and **SGet**, which we have added as a common extension to both the MRS and MPS systems.

- (**Show** *presentation-surface*): Transfer control to a given presentation surface, refresh the form involved, handle cursor movement and user input.
- (**SSet** *presentation-surface object-type object-unique-id Slot-name value*): Set a slot in a record by tracing the path given by the argument list.
- (**SGet** *presentation-surface object-type object-unique-id Slot-name*): Get a slot value, again by tracing the given path.

If the user enters several values on a form, these values are stored directly in the MPS database and the values are sent to the inference process as a list of ordered triples of the form "(input-slot form value)". Each such triple can be stashed without modification into the knowledge base, since the MPS representation (form slot value) directly corresponds to the MRS representation of (property component value). For example, if the inference engine stashes a value as a result of some inference, such as

(**stash** '(prf f1 12.5))

then a procedural attachment is activated, which stashes the value into the knowledge

base and initiates the following events:

- Determines whether "(prf f1 12.5)" is isomorphic to one of several legal patterns (in this case "(slot form value)") and, if so, sets slot to prf, form to f1, and value to 12.5.

- Invokes the remote MPS call "(SSet 'f1 'input '(prf f1) 'value 12.5)", which tells the presentation surface "f1" to set its "prf" slot to "12.5". MPS alters the display accordingly.

The MPS primitives always perform a last evaluation on their arguments before sending themselves as a message to the inference process. This allows the MRS assertions to be of arbitrary complexity so long as their structure is isomorphic at the top level to one of the accepted patterns.

MAINTAINING CONSISTENCY: THE COP COMMUNICATION AND CONTROL SYSTEM

The inference and user interface processes contain data that are largely distinct semantically: rules are under the purview of the inference process, while the details of the presentation surface are the domain of the user interface alone. The two processes do, however, experience overlap in several semantic domains, most notably for (a) data entered by the user and (b) inference results that are to be presented to the user. Because multiple copies of these data exist in the two name spaces and because of their time-varying nature, the potential exists for inconsistency between the two databases. For example, the inference process could erroneously work on obsolete data if newly entered data held in the user interface database are not yet installed in the inference process's knowledge base; similarly, incorrect results (or none at all) could be displayed to the user even after the inference process had determined the proper values, if the corresponding entries in the user interface database have not been updated.

We need not solve all the problems addressed by a distributed DBMS such as SDD-1 [6], since only one data source (the user or expert system) can be updating the database at any moment in time. ASTA is, however, faced with the problems of maintaining multiple copies of a single logical database that a distributed DBMS must solve. In order to ensure consistency and integrity of the common data across the two processes, ASTA employs a *time-stamping* data communication architecture with message routing between the two processes explicitly managed by a semi-intelligent controller. The time-stamping, message-passing architecture from which ASTA has been constructed is the COP system [4].

In its full generality, the COP communications and control system provides for resource management and planning functions as well as communications; within the ASTA implementation, however, we use it primarily to ensure timely and reliable interprocess communications. COP provides to each of the two client processes (user interface and inference) a message-passing view of the external computational environment. All communication between modules is effected by means of a single **send-message** primitive having the following structure:

(**send-message** *from to timestamp class* [*text*])

where *from* and *to* identify the sender and receiver, *class* is a member of the set of permissible remote operations which the sender may invoke (in the case of ASTA, the set {**SSet, SGet, Show**}), and *timestamp* is a structured message identifier that includes a timestamp timed according to the sender's clock and guaranteed to be unique across all clients in the system. Where messages take argments, *text* specifies those arguments, in keyword (attribute-value) form. (The *from* value is provided by the communications slave, described below, to minimize the potential for implementation errors on the part of the client module's designer and to prevent message forgery.)

Each client is transparently provided with a communications slave that performs two functions:

- It dispatches messages sent by the client using **send-message**.
- It responds to incoming messages, either directly or by dispatching them to the appropriate client function.

If an incoming message is a member of the client's set of permissible messages, message receipt is transformed into a call to the client's corresponding function. If the message is not a member of the client's operator set, the slave determines whether it is an otherwise known message class (such as a bookkeeping request from the controller which the slave itself can execute); if not, an error message is transmitted back to the controller by the slave.

As described above, our approach to ensuring consistency for inferred facts and hypotheses relies upon *procedural attachments* within the inference process. These procedural attachments are essentially explicit per-rule and per-relation specifications of the corresponding relation(s) in the companion database that depend upon the value of the attached rule or relation. For example, the inference engine is instructed to check for and execute procedural attachments whenever a new fact is inferred in order to ensure immediate update of the user interface database.

Depending upon the specifics of the rule or relation so attached, the procedural attachment may specify either a *remote procedure call* form of update, wherein the

inference engine waits until the companion database has been updated before proceeding, or a pure *message delivery update*, in which messages are queued for delivery to the companion process but the sender does not wait for delivery. (In the latter case a remote procedure call is guaranteed to wait until all pending update messages have been delivered and acknowledged.)

Emulation of remote procedure call is supported by allowing the sender to "block" itself until a return message is received; this blocking action is a primitive of the communications slave, which continues to listen to the input port and process incoming messages, either queueing them on an agenda of tasks to feed the client once it is resumed or executing them directly if they are communications-specific (such as a request for a message indicating the client's status). This restricted use of message passing permits two clients to synchronize their state: for example, the sender client may ask for a datum from the target client's database and then wait, blocking all other analysis- and display-related operations within its name space until a (time-stamped) return value is received.

CONVERSING WITH AN EXTERNAL DATABASE

A wide variety of data bases are already in use by radar system designers and analysts, and clearly ASTA will benefit if it can capitalize on the prior existence of massive collections of relevant domain data. Unfortunately, these data bases are mutually incompatible, are frequently poorly organized, and are supported on a variety of DBMS's. An important task in the design of ASTA has therefore been the development of a uniform method of providing the inference process with access to external database systems.

Our solution to this practical problem has been to isolate the knowledge about specific external database query languages and schemas in a single additional *database access expert* process that maps data requests from the inference engine to the appropriate database query. This prevents the inference process's knowledge base from becoming cluttered with arcane knowledge of the external DBMS, provides a potential degree of parallelism (in that inference can proceed while the database access expert formulates a database query), and modularizes the DBMS-specific knowledge so that no changes to the inference process's knowledge base are necessary (at least in principle) in order to support access to additional external databases or to change the query protocol as external databases evolve.

While this meets our immediate pragmatic goal of providing access to external DBMS's, several problems still present themselves during the construction of the database access expert. For example, a separate access capability (whether in a monolithic access expert or multiple indpendent such experts) must be provided for each external DBMS. Furthermore, the schema of the external DBMS must be duplicated in the database access expert, and thus changes in the external database schema still require a corresponding alteration of the database access expert. Finally, the database

access expert must contain knowledge not only of the query language syntax, but of the computational semantics of query language constructs as well, in order to effectively map inference process requests into database queries. Whereas this semantic mapping is normally performed either by a human user of a conventional database or by a transaction designer, the database access expert cannot appeal to either source of interpretive expertise and thus must carry the additional burden of maintaining and applying that knowledge of query semantics itself.

On the basis of our experience with the expert system–external database interface, we observe that a more effective solution is at hand if data base systems provide an additional query capability not currently supported: that of *query predicate* satisfaction. That is, an ideal external database from the standpoint of the expert system would be one that accepts not merely a static query but a predicate that can be executed within the name space of the database.

Such a query predicate could contain, for example, a weighted vector of values or ranges which would be applied to candidate tuples in the database by the DBMS to produce a *degree-of-match* measure, where a database tuple is considered to itself be a vector in n-space. In the simplest case, attributes of a relation would be objects with considerable mathematical structure on them (such as values in $\mathbf{R}^n$ with the elements of the basis set being semantically compatible -- for example, a database that contained only latitude and longitude information), and the metric for degree of match is little more than a variance calculation. In more difficult cases where, for instance, the range of data values for an attribute is nominal-level (i.e. no ordering relation applies), the query predicate must contain more information in order to convey the *goal* of the "user" (in this case, the inference engine) in posing the query. For instance, if the attribute in question is a spectral color, the degree to which "yellow" is a satisfactory match for a query that requested tuples that are "like orange" is a function of the intent of the inference engine in using that color information, which intent must be reflected in the query predicate's handling of the color attribute. This predicate would then allow the database to perform a better-informed search (i.e. provide better recall performance) by virtue of semantically-derived information provided as a part of the query by the agent (the inference engine) formulating that query.

An obvious extension of this approach, and one especially useful in expert system applications, would be a DBMS that provides not only the tuples that match above some threshold or according to semantic constraints embodied in the query predicate, but also the degree-of-match measure as defined by the query function for each tuple satisfying the query predicate.

A database system that supported query predicates would offer a potential performance improvement in several respects: raw query-satisfaction speed, decreased inter-process traffic, and decreased requirements for post-processing of query results by the inference engine, all by virtue of the knowledge contained in the query that focuses the database search process. Such cooperation between the expert system and the external database would greatly facilitate the development of knowledge-based problem solvers in practical problem domains. We know of no database systems that currently

provide such a facility.

SUMMARY AND CONCLUSIONS

We have described a knowledge-based consultation system which uses a combination of AI inference techniques and distributed data base management. In developing this system we have been able to provide workable solutions to the merging of a Predicate Logic based inference engine and a relational data base. This required both the maintenance of copies of the same data in two quite different representations, as well as the development of efficient mechanisms by which the two data bases remain consistent. This would not be new or interesting if these were simply two data bases. What is significant is that we were able to utilize techniques quite similar to the time-stamping approach of SDD-1 with one copy of the data in a DBMS and the other in a predicate logic based system.

An approach has also been incorporated into ASTA which allows the knowledge-based system to access and query external DBMS's. The approach allows the inference engine to generate queries to the data bases in a predicate logic form, and to have these queries transformed into appropriate DBMS queries to the external data base. The results of the query are then used to instantiate the predicate logic assertions which are delivered back to the inference engine as if they were obtained directly by the inference engine. We have proposed an extension to this technique which would enable the inference engine and the DBMS to more closely cooperate in the selection of data with an imprecise query. This is an important problem which needs to be solved for many DBMS/Inference-Engine interactions in support of knowledge-based systems.

ACKNOWLEDGMENTS

The authors wish to acknowledge the efforts of the many people who have contributed to the ASTA project, especially: Victor Askman, John L. Allen, Eric Domeshek, Ellen Drascher, Nancy English, Milissa Feeney, Brian P. McCune, Sonia Schwartzberg, David S. Spain, Richard P. Wishner, and James L. Whitaker.

REFERENCES

[1] Wilson, G., Domeshek, E., Drascher, E., and Dean, J. (1983). The Multipurpose Presentation System. *Proc. Ninth International Conference on Very Large Data Bases.* Florence, Italy, October 1983.

[2] Adams, T., Cromarty, A., McCune, B., Wilson, G., Grinberg, M., Cunningham, J., and C. Tollander (1984). A Knowledge-Based System for Analyzing Radar Systems, invited paper, Proceedings of Military Microwaves '84, London, England.

[3] Genesereth, M. (1982). An introduction to MRS for AI experts. Technical Report HPP-80-24, Stanford University Heuristic Programming Project, November 1982.

[4] Cromarty, A. (1984). COP: A framework for the intelligent Control of Processes using Communication over Ports. (In preparation.)

[5] Codd, E. (1979). Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems* 4(4):379-434.

[6] Rothnie, J., Bernstein, P., Fox, S., Goodman, N., Hammer, M., Landers, T., Reeve, C., Shipman, D., and Wong, E. (1980). Introduction to a system for distributed databases (SDD-1). *ACM Transactions on Database Systems* 5(1):1-17.

APPENDIX B: RUBRIC OVERVIEW

This appendix contains a paper on the RUBRIC system presented at the *8th Int. ACM-SIGIR Conf. on Research and Development in Information Retrieval.* Montreal, June 1985.

RUBRIC
An Environment for
Full Text Information Retrieval*

Richard M. Tong
Victor N. Askman
James F. Cunningham
Carl J. Tollander

December 21, 1984

Advanced Information & Decision Systems
201 San Antonio Circle, Suite 286
Mountain View, CA 94040.

Abstract. This paper describes an ongoing research project that is concerned with developing a computer-based aid for retrieval from unformatted full text databases. Unlike other attempts to improve upon Boolean keyword retrieval systems, our research concentrates on providing an easily used rule-based language for expressing retrieval concepts. This language draws upon work both in artificial intelligence and the theories of evidential reasoning to allow the user to construct queries that give better precision and recall than more traditional forms. The paper includes a discussion of our formal model of retrieval, the rule language, and some comparative performance results.

Keywords. Information Retrieval, Artificial Intelligence, Expert Systems, Evidential Reasoning, Fuzzy Sets.

*To appear in:
Proc. 8th Int. ACM SIGIR Conf. on R&D in Information Retrieval.
Montreal, 1985.

1. INTRODUCTION

This paper describes an ongoing investigation into the application of ideas from Artificial Intelligence (AI) in the development of a computer-based aid for Information Retrieval (IR). The prototype system, called RUBRIC, is designed to help IR professionals gain easy access to large unformatted full text databases. Knowledge about retrieval requests is encoded in RUBRIC as a collection of rules with attached uncertainty values. This representation provides the framework for an appropriately expressive query language that can represent partial relevance and which by its modular nature is easily understood and modified. When coupled with an effective user interface, the rule-based approach can, we believe, give significant improvements over commercially available Boolean keyword systems such as DIALOG, LEXIS, and MEDLARS. At the same time, it avoids the theoretical and computational problems associated with full scale natural language processing of documents (e.g., as proposed by Lebowitz, 1983), and the difficulties users have in understanding the mechanisms used in statistical approaches (e.g., Salton's SMART system, 1971).

RUBRIC differs in several important ways from traditional approaches, namely in that:

- Matching is performed over the whole document, rather than in certain predefined fields.
- Document retrieval is not two valued since documents can be given a relevance value in the range [0,1].
- Queries are expressed in a language of rules that allows the user to develop hierarchical knowledge structures of retrieval concepts.
- Users are provided with a collection of tools to help develop and modify queries, and to analyze the retrieval results.

In providing these characteristics RUBRIC makes use of several key ideas from AI. In particular, RUBRIC is an example of a production system that can perform evidential reasoning. In this view, the text of the document is the "evidence" on which the system determines the relevance of that document to the retrieval request. The knowledge on which this judgement rests is embodied in the rules which link retrieval concepts. In contrast to conventional expert systems, the knowledge is entered directly by the user of the system who thereby acts as his or her own expert. In addition, RUBRIC makes use of an object oriented presentation system to provide the necessary flexibility at the user interface.

2. THE RETRIEVAL MODEL

In developing our model we start from the premise that the function of a retrieval system is to select a sub-set of the documents in the database as defined by their *relevance* to the user's query. The inherent imprecision in the concept of relevance requires that this be a *fuzzy* sub-set (Zadeh, 1965) rather than a classical one. Suppose, then, that the user has a finite set of retrieval concepts, $\boldsymbol{C}$, of interest:

$$\boldsymbol{C} \triangleq \{c_1, c_2, \cdots, c_M\}$$

and that the database, $\boldsymbol{S}$, contains a finite number of documents:

$$S \triangleq \{s_1, s_2, \cdots, s_N\}$$

then there is a fuzzy relevance relation, R, from C to S such that:

$$R(m,n) = \textit{the relevance of document } s_n \textit{ to concept } c_m$$

If we now assert that relevance can be quantified as a real number in the interval [0,1], then for any particular concept, c, which is an element of C, we can extract from R a row-tuple, $R^*(c)$, which then defines a fuzzy sub-set of S. This sub-set is the ground truth against which we wish to measure the performance of our retrieval system. Our goal therefore is to build a system that can generate an $R(c)$ which accurately "estimates" $R^*(c)$; with an ideal retrieval system giving $R(c) = R^*(c)$ for every c in C

To make our fuzzy set theoretic interpretation of the IR problem operational we need a calculus for the representation and propagation of relevance values. Since we assume that relevance can be represented as a numerical value in the interval [0,1], and that there is an obvious fuzzy set theoretic interpretation of these values, we can draw upon work on many-valued logics (Rescher, 1979), and on the use T-norms as models of fuzzy set intersection (e.g., Dubois and Prade, 1982), to help us construct a calculus of relevance values.

The first task is to define a set of operators for conjunction (the *and* connective), and disjunction (the *or* connective). There are many we could choose, but we shall consider four pairs as shown in Table 1. Here $v[A]$ and $v[B]$ denote the relevance values of the primary propositions, with $v[A \textit{ and } B]$ and $v[A \textit{ or } B]$ denoting the relevance value of their conjunction and disjunction respectively. The conjunction operators are T-norms, the disjunction operators are T-conorms, and the negation (the unary operator *not*) is defined by $v[\textit{not } A] = 1 - v[A]$.

The second task is to define a mechanism for performing rule-based inference. In two-valued logic the *modus ponens* syllogism allows B to be inferred from A and $A \Rightarrow B$. In an infinitely-valued logic, we need to extend this idea so that the relevance of B, denoted $v[B]$, can be computed from any given $v[A]$ and $v[A \Rightarrow B]$, where "$\Rightarrow$" is some infinitely-valued implication. Functions that allow us to compute $v[B]$ are called detachment operators (and are denoted $*$). It is usual to define them so that for a given definition of $\Rightarrow$, $v[A] * v[A \Rightarrow B]$ is a lower bound on the value of $v[B]$. Five of these are shown in Table 2, together with the corresponding implications.

Let us denote a particular calculus by L(i,j) where "i" is an index over the conjunct-disjunct operator pairs and "j" is an index over the detachment operators. Then we see that some of the L(i,j) are well known; in particular, L(3,3) is Lukasiewicz's nondenumerably infinite logic (Lukasiewicz, 1930), and L(3,0) is a logic proposed by Zadeh (1973). Another calculus of interest is L(2,2), which we can view as a "pseudo-probability" logic in which A and B are independent events.

The way in which RUBRIC generates R(c) is to interpret the rules as a hierarchy of retrieval concepts and sub-concepts. Thus by naming a single concept, the user automatically invokes a goal oriented search of the tree defined by all of the sub-concepts that are used to define that concept. The lowest-level sub-concepts are

Table 1 Conjunct-Disjunct Operators

	v(A and B)	v(A or B)
0	T[v(A),v(B)]★	S[v(A),v(B)]★
1	max[0,v(A)+v(B)-1]	min[1,v(A)+v(B)]
2	v(A).v(B)	v(A)+v(B)-v(A).v(B)
3	min[v(A),v(B)]	max[v(A),v(B)]

★T[1,1] = 1, T[x,1] = T[1,x] = x,
T[x,y] = 0 $\forall x,y \in [0,1)$

★S[0,0] = 0, S[x,0] = S[0,x] = x,
S[x,y] = 1 $\forall x,y \in (0,1]$

Table 2 Detachment and Implication Operators

	Detachment (*)	Implication (=>)
0	v(B) = min[v(A),v(A=>B)]	min[v(A),v(B)]
1	v(B) = min[v(A),v(A=>B)] if v(A)+v(A=>B) > 1 = 0 otherwise	max[1-v(A),v(B)]
2	v(B) = v(A).v(A=>B)	min[1,v(B)/v(A)]
3	v(B) = max[0,v(A)+v(A=>B)-1)]	min[1,1-v(A)+v(B)]
4	v(B) = max[0,(v(A)+v(A=>B)-1)/v(A))]	1-v(A)+v(A).v(B)

themselves further defined in terms of pattern expressions in a text reference language which allows keywords, positional contexts, and simple syntactic and semantic notions. The relevance values attached to each rule then provide, together with an appropriate calculus of relevance values, a mechanism for determining the overall relevance of a given document as a function of those patterns which it contains.

3. THE RULE LANGUAGE

RUBRIC is a system that uses a rule-based reasoning process and in this section we describe the nature of a rule and its constituent parts. In Figure 1 we indicate the most general form of rule that can exist within the system. Its two parts consist of the primary inference which links the primary antecedent to the consequent concept, and the secondary inference which describes how the auxiliary antecedent modifies the

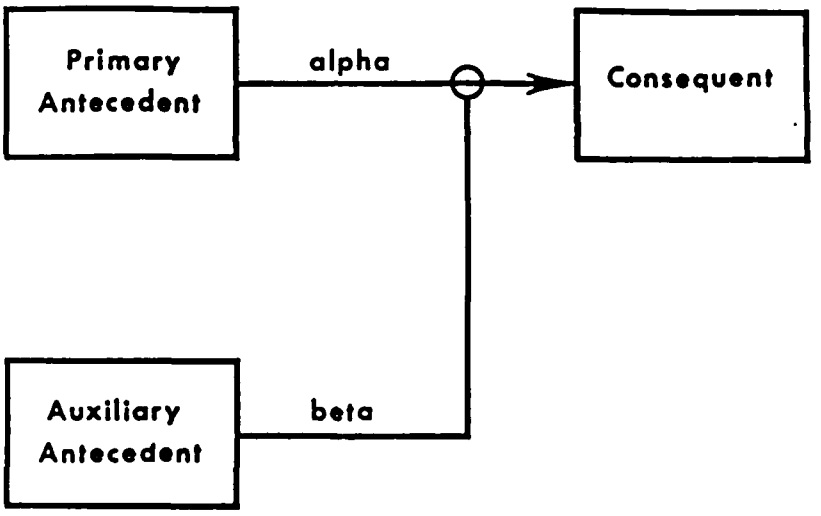


Figure 1 The General RUBRIC Rule

primary inference. The motivating idea behind the secondary inference is that there are cases in which the existence of additional evidence would cause us to modify our original inference, with the proviso that this auxiliary evidence is by itself of no direct interest. Our new rule form models the effect of such evidence by changing the weight attached to the primary inference.

Formally we model this as:

$$v\,[consequent\,] = v\,[primary\_antecedent\,] * v\,[rule\,]$$

with

$$v\,[rule\,] = \{alpha + (beta - alpha) \times v\,[auxiliary\_antecedent\,]\}$$

where alpha and beta are the relevance values associated with the primary antecedent and the auxiliary antecedent respectively, * denotes an appropriate detachment operator, and $v\,[\cdot]$ denotes the relevance value of a variable. Notice that we have chosen to model the effect of the auxiliary antecedent by a simple linear interpolation function. Given our current understanding of the impact of this rule form it seems to be an appropriate choice. Notice too that we allow at most one secondary antecedent. In the future may want to allow multiple secondary antecedents and will then have to define mechanisms to deal with conflict resolution.

3.1. Rule Types in RUBRIC

We provide a variety of rule types in RUBRIC. They have similar syntactic and functional forms but their semantics are intended to capture the different types of inferential relations that can exist between retrieval concepts. That is, we provide a rule language that allows the user of RUBRIC to express the required relationships between the concepts of interest. Since the rules carry semantic information they can be used to help perform more efficient searches of rule trees.

We briefly discuss the five rule types and then consider the elements used in constructing antecedent expressions:

The IMPLIES rule. This is the principal rule type implemented in RUBRIC. It is intended to link retrieval concepts and then be invoked in a generalized *modus ponens* inference. That is:

$$v\,[b\,] = v\,[a\,] * v\,[IMPLIES\,(a\,,b\,)]$$

where * is the appropriate detachment operator. Note that $v\,[IMPLIES\,(a\,,b\,)]$ is given as part of the rule definition (i.e., it depends on the values of alpha and beta and is given by the expression in braces in the definition above) whereas $v\,[a\,]$ is derived by the system from the application of other rules.

The EVIDENCE rule. This rule type is used to link text references to concepts. It is intended to capture the notion that text expressions are used as direct "evidence" in determining the relevance of the document to the retrieval topic. Functionally, this rule is similar to IMPLIES but we want to distinguish between inferences made using EVIDENCE and IMPLIES so as to provide better control of search. We have:

$$v[b] = v["a"] * v[EVIDENCE("a", b)]$$

where * is a detachment operator, and "*a*" is a text reference expression.

The SUBSET rule. This rule type allows us to express the relationship between a subset of a set and the set itself. It seems that these rules perform no modification of relevance values, but the reason we introduce them is to allow ourselves to take account of the length of the reasoning chain used to establish the relevance of a document.

The INSTANCE rule. A rule that allows us to express the relationship between an element of a set and the set itself. As with the SUBSET rule, there seems to be no need to modify relevance values.

The ATTRIBUTE rule. This rule is intended to capture the idea that concepts have components (or attributes), and that knowledge of these components may be used to help establish the presence of the concept itself.

3.2. Antecedent Operators in RUBRIC

These operators are the primitives used in developing the primary and secondary antecedent expressions. There are three main classes: (1) those which take concepts and text as arguments (i.e., the "logical" operators), (2) those which take only text (i.e., the "distance" and "boolean" operators), and (3) those which perform miscellaneous functions (i.e., the "scope" operators, the "proximity" operator and the "macro" function).

The Logical Operators. These operators take either concepts or text, or both, as their arguments, which themselves can be arbitrarily complex expressions using other antecedent operators. We allow generalized (i.e., multi-valued) forms of AND, OR, NOT, as well as two non-traditional operators BEST-OF and WEIGHT-OF which capture the idea that (1) any one of the arguments would be appropriate so we might as well take the best, and (2) the more arguments that are true the better.

The Distance Operators. These operators take a pair of text arguments and return a value which represents the distance between them. Currently we have implemented three fuzzy operators, and two boolean operators that also double as scope operators. The NEAR_W, NEAR_S and NEAR_P operators all return a value in the interval [0,1] which is a normalized measure of the distance in words, sentences or paragraphs between its arguments. The SENTENCE and PARAGRAPH operators each take a pair of keywords as arguments and tests to see if they occur within the same sentence or paragraph in the document.

The Boolean Operators. These operators take only text as their arguments and return a value from the set {0,1}. The PRECEDES operator takes two keyword arguments and tests whether one occurs before the other. The WITHIN operator takes two keyword arguments and tests whether they are within some distance (in words) of one another. The PHRASE operator takes multiple text arguments and tests whether the phrase defined by concatenating the keywords occurs within the document.

The Scope Operators. In their most general forms these operators are somewhat problematic. Conceptually they are straightforward, but their implementation is complicated. The SENTENCE and PARAGRAPH operators mentioned above are degenerate examples and are more conveniently thought of as distance operators with discontinuous functional forms. Scope operators take only one argument and their intended effect is to reduce the scope of the pattern matching to the scope unit indicated. Notice that there is an implied default scope unit of "document" if no scope operator appears. Obviously, there are some constraints on the way these operators can be nested. We allow scoping using two functions. The *SENTENCE* operator reduces the scope of the pattern matching to a single sentence. The argument can be any expression of antecedent operators and concepts and text. Similarly, the *PARAGRAPH* operator reduces the scope of the pattern matching to a single paragraph.

The Proximity Operator. This operator allows the user to take account of the "nearness" of concepts within a document. This is still an experimental feature and we are exploring the semantics of concept location together with appropriate distance measures.

The Macro Function. This is a feature that allows the rule writer to enter a special synonym symbol in any place where a text string could appear. When RUBRIC encounters such a symbol it recognizes it as a "place holder" for a set of synonymous text strings and expands the language expression accordingly.

3.3. Aggregation Functions

These functions determine how we will combine the inferred relevance values from multiple rules having the same retrieval concept as their consequent. In the current implementation we have an implied OR (i.e., a disjunction of the evidence), although the AGGREGATION function can be specified independently of any choices we make for the other operators. We also provide for alternative AGGREGATION functions, to be implemented as we consider the effects on nodes with multiple types of rules.

4. THE USER ENVIRONMENT

The target machine for RUBRIC is a professional workstation, such as a SUN, with high resolution bit-mapped display capabilities. In order to exploit the graphics facilities on such machines we have designed the user interface for RUBRIC around a multi-purpose interface system called MPS (Wilson *et al.*, 1983). MPS utilizes object oriented descriptions of the information to be exchanged between the user and RUBRIC, thus allowing a clean demarcation between the functions of RUBRIC and those of the user interface. In addition to data about specific objects to be displayed, MPS maintains generic descriptions of the contents of presentation surfaces, so employing a high-level semantic model of the objects to be displayed. The MPS interface uses a relational database as the medium of communication, and since the semantic model is stored directly in this relational database, it is available to both RUBRIC and MPS. An important benefit of the cleanly defined interactions between RUBRIC and MPS is that

RUBRIC is freed from the details of handling numerous user interface devices.

The current version of the interface supports a menu driven style of interaction on conventional alphanumeric terminals. Selection of menu items is done by positioning a cursor via function keys, input of information is done through displayed forms, and presentation of data is done by means of non-overlapping windows. An example of a RUBRIC information entry form is shown in Figure 2. This is the new rule template form. The user can call it from a menu and is then expected to enter the appropriate information in the outlined areas. A screen editor with built-in syntax checking is provided. Notice the provision of a sub-menu within this form; the user can ask for help if necessary, and is provided with a mechanism for exiting when he or she is satisfied with the new rule.

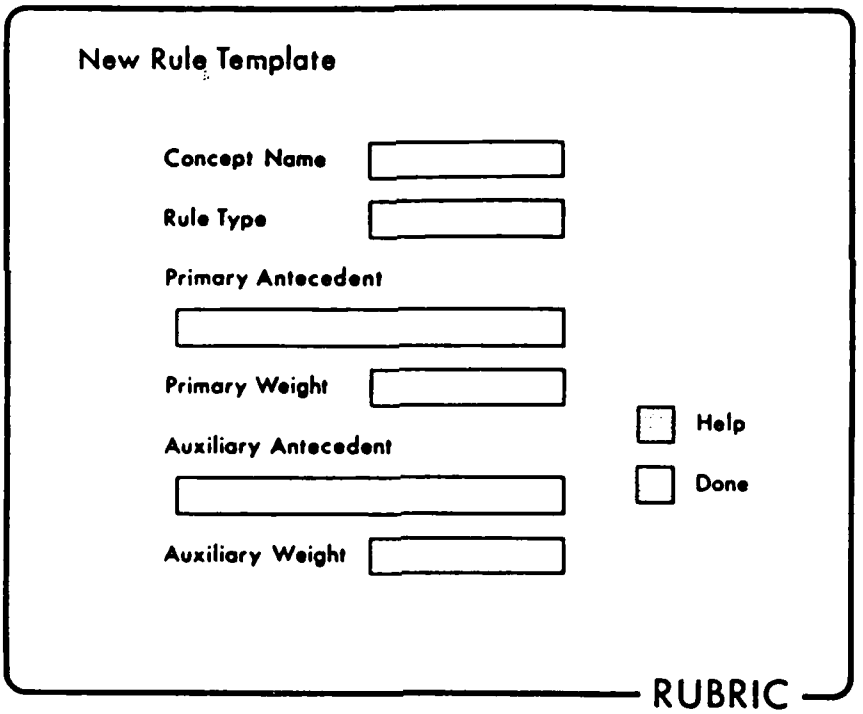


Figure 2 New Rule Template Form

Using MPS allows us to provide effective interaction mechanisms so that the iterative and incremental nature of query building can be properly supported. Our view is that the user first constructs an initial query by drawing upon existing knowledge in the rule-base and perhaps adding small amounts of additional knowledge. Having done that, he or she checks over the query for obvious flaws and errors, and then applies it to a document database. Most probably, the initial test of the new query will be on a small database of documents with which the user is familiar. Using the results of the test as a guide the user will make appropriate changes to the query and repeat the cycle. When the performance is satisfactory, the query will be applied to the main document database(s). If the query represents a retrieval concept that the user is likely to want to use again, then it can be entered into the permanent rule-base.

Given that query building passes through these stages then we can see the need for several categories of tools to support query development. These tools are really not tools for knowledge elicitation *per se*, but rather tools that will provide an "environment" in which the user can develop queries easily, get useful feedback about their performance and quickly make changes if this performance is not satisfactory. By analogy, we think of this "toolbox" as a collection of tools that either singly or in combination can help the user perform some activity.

Let us consider the tools that we provide in a little more detail:

Query Construction Tools. These tools are needed to help the user develop and edit new queries. They include mechanisms for describing the retrieval concept in terms of the knowledge in RUBRIC's existing rule-base(s), an editor and a syntax checker.

Rule-Base Access Tools. The purpose of these tools is to allow the user to explore the rule-base. For example, a user who is about to develop a new query many want to browse the rule-base to see if similar retrieval concepts already exist or to examine sub-concepts that might form a basis for the new extended query.

Static Check Tools. These tools might be invoked either by the user or by the system to check that a newly create query is "consistent." For example, such tools check that there are no circular paths in the reasoning, that the query can indeed return a relevance value significantly different from zero, etc.

Performance Analysis Tools. The user needs to be able to examine the results of the retrieval request in a variety of ways. A minimum requirement is for the results to be displayed in graphical as well tabular form. The user will also certainly want to be able to compute a variety of performance measures based on our fuzzified notions of precision and recall. He or she will also need some mechanism for storing performance results so that subsequent modifications to the query can be compared.

Diagnosis Tools. We expect that a large part of the knowledge elicitation process will be concerned with trying to understand *why* the query performed the way it did. To support this type of activity we provide a class of tools that allow the user to do things such as single-step the query, explore the sensitivity of the query to changes in its structure, generate traces of rule invocation, create artificial documents for checking sub-parts of the query and observe bottom-up propagation of user induced triggering of

rules.

Help. Finally, we provide a generalized help facility within RUBRIC. At any stage in the process the user should be able to ask for on-line help that would explain the general features of the RUBRIC system, the purpose of a tool, or the nature of the response required at a decision point.

5. SYSTEM IMPLEMENTATION

RUBRIC is currently implemented in a Berkeley UNIX[1]/VAX 11-780 environment. The implementation is divided into two major modules: the preprocessor module and the system module. The preprocessor module is written in the C Language and takes as input the free format text of a collection of documents and builds the RUBRIC-readable database for that collection of documents. The primary component of this database is an inverted structure on the words (actually, word stems) occurring in the collection of documents. Each word has one entry in the structure and is accompanied by various contextual information (such as in which document(s) and in which position(s) it occurs). The system module, which includes the user interface, the toolbox and the retrieval subsystems, is currently implemented in both Franz LISP and C. In general, the lower level word matching/database access functions are implemented in C, while the higher level query expansion/tree traversal functions are implemented in Franz LISP.

6. RETRIEVAL PERFORMANCE

We have performed a variety of experiments with RUBRIC to assess its effectiveness. These include tests of the impact of using different uncertainty calculi and restrictions on the use of the rule language (Tong and Shapiro, 1984), as well as timing and sizing tests. However, to illustrate our methods we will describe some experiments that were designed to assess the improvements that can be achieved over a conventional Boolean keyword approach.

As an experimental database for testing the retrieval properties of RUBRIC, we have used a selection of thirty documents taken from the Reuters News Service. Our basic experimental procedure is to rate the documents in the database by inspection (i.e., define the relevance relation row-tuple $R^*(c)$), construct a rule-based representation of a typical query, apply the query to the database, and then compare the rating, $R(c)$, produced by RUBRIC with the *a priori* rating $R^*(c)$.

Given our fuzzy set interpretation of the IR problem, there are a large number of possible measures of performance that we could employ. For this presentation we concentrate on just two. Both of these are based on the idea of using a selection threshold to partition the ordered documents so that those above it are "relevant" (either fully or marginally) and those below it are "non-relevant." In the first we lower the threshold until we include all those deemed *a priori* relevant, and then count the number of unwanted documents that are also selected (denoted N_F). In the second we raise the threshold until we exclude all irrelevant documents, and then count the number

[1] UNIX is a Trademark of AT&T Laboratories

of relevant ones that are not selected (denoted N_M). The first definition therefore gives us an insight into the system's ability to reject unwanted documents (precision), whereas the second gives us insight into the system's ability to select relevant documents (recall).

We selected as a retrieval concept "Violent Acts of Terrorism," and then constructed an appropriate rule-based query. This is summarized in tree form in Figure 3, where we make extensive use of the extended rule form described above. An auxiliary_antecedent is shown linked to a primary inference by a horizontal directed arc. Application of this query to the document database with calculus L(3,2) (i.e., one that models conjunction/disjunction as min/max and detachment as product), results in the document profile shown in Figure 4. (Notice that for presentation purposes the relevance scores have been re-normalized and the documents ordered such that those determined to be *a priori* relevant are to the left in Figure 4.) This is excellent performance of course; the relevant and non-relevant documents being correctly partioned into two disjoint sets. (e.g., setting the selection threshold at 0.3 would make N_F and N_M simultaneously zero.)

To compare RUBRIC against a more conventional approach, we constructed two Boolean queries by using the rule-based paradigm and setting all rule weights to 1.0. (Thus showing, incidentally, that our method subsumes Boolean retrieval as a special case.) One of these queries is shown in Figure 5 as an AND/OR tree of sub-concepts. The only difference between the two Boolean queries is that in the first we insist on the conjunction of ACTOR and TERRORIST-EVENT (as shown), whereas in the second we require the disjunction of these concepts. Running each of these queries against the thirty document Reuters database produces a non-fuzzy subset of documents. Performance is then assessed in the conventional way with *recall* computed as the ratio of the number of relevant documents retrieved to the total number of relevant documents in the database, and *precision* computed as the ratio of the number of relevant documents retrieved to the total number retrieved. To get an equivalent RUBRIC score we construct a non-fuzzy set from R(c) by setting the relevance threshold and then marking as " retrieved" all those documents with higher relevance values, and "not-retrieved" all those with lower values. The conjunctive form of the Boolean query misses five relevant documents and selects one non-relevant document, giving:

$$\textit{Precision} = .89 \qquad \textit{Recall} = .62$$

The disjunctive form selects all the relevant documents, but at the cost of also selecting seven of the non-relevant ones, giving:

$$\textit{Precision} = .65 \qquad \textit{Recall} = 1.0$$

However, if we select the relevance threshold to be 0.3, then the RUBRIC retrieval gives:

$$\textit{Precision} = 1.0 \qquad \textit{Recall} = 1.0$$

While these results represent only a partial test, we believe that they indicate that the RUBRIC approach allows the user to be more flexible in the specification of his or her query, thereby increasing both precision and recall. A traditional Boolean query

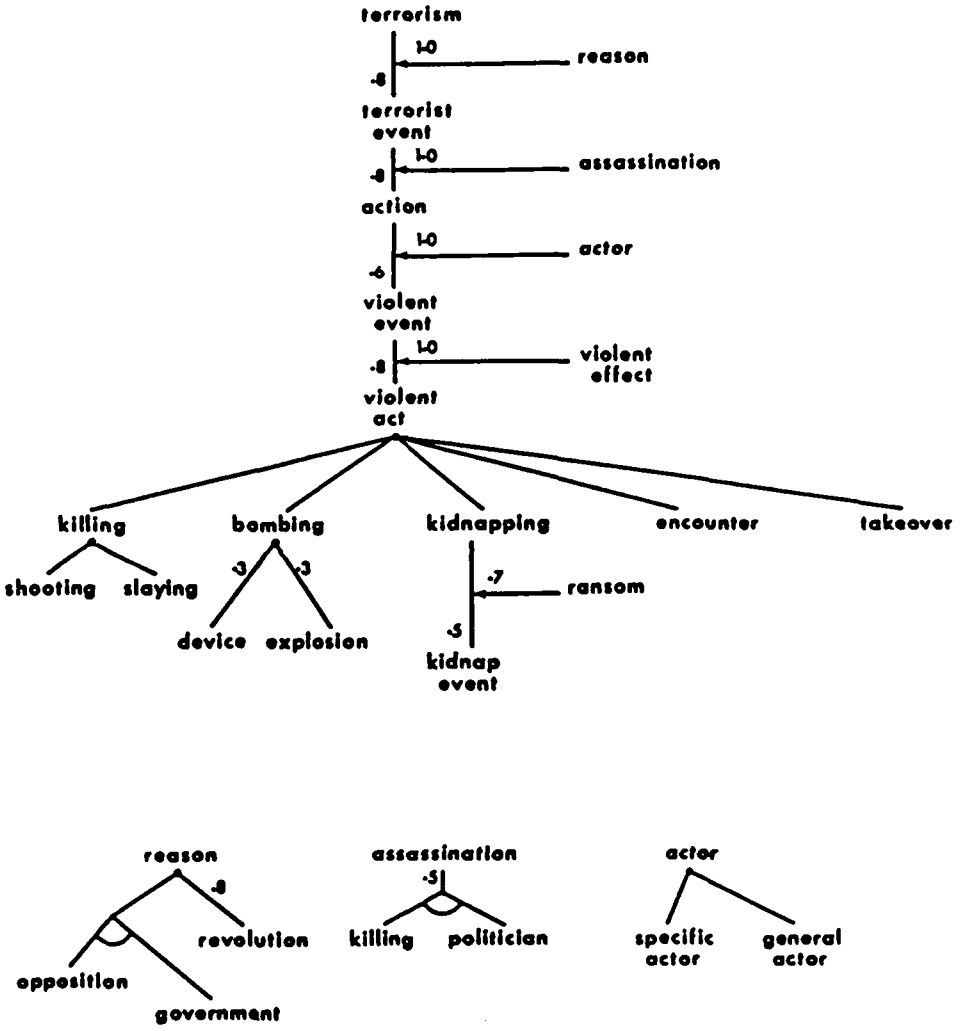


Figure 3 Example RUBRIC Query Tree

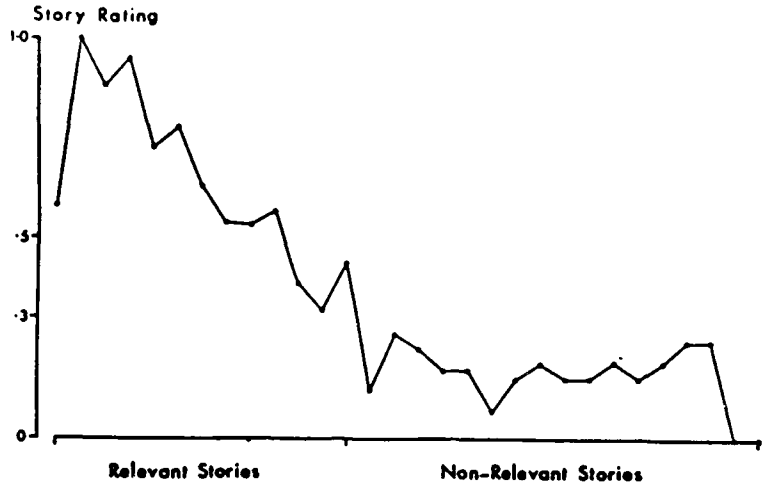


Figure 4 RUBRIC Retrieval Profile

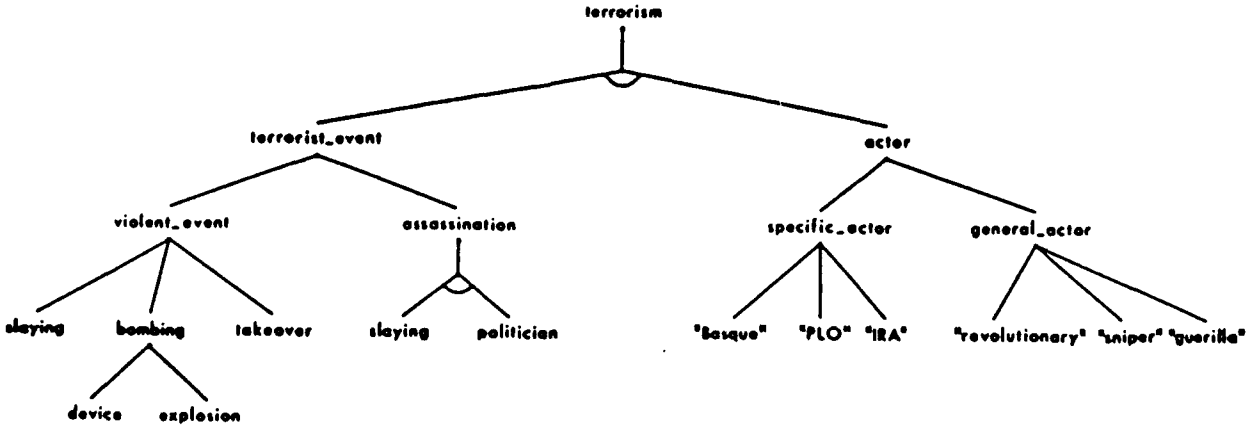


Figure 5 Comparative Boolean Query Tree

tends either to over-constrain or under-constrain the search procedure, giving poor recall or poor precision. We feel that, given equal amounts of effort, RUBRIC allows better models of human retrieval judgment than can be achieved with traditional Boolean mechanisms.

7. SUMMARY

In this paper we have attempted to give an overview description of RUBRIC and the ideas on which it is based. Although it is still a research prototype we believe it shows considerable promise as an advanced IR system.

We believe that the major contributions of RUBRIC are that it encourages proper structuring of queries leading to more effective and better understood retrievals. Given equal amounts of effort, RUBRIC can give improved precision and recall when compared to conventional systems. Further, the provision of an advanced interface and toolbox gives the user an environment in which the IR task can be performed quickly and effectively. Finally, because of its inherent modularity, RUBRIC is an excellent vehicle for exploring a wide range of related research issues such as the problem of the representation and manipulation of uncertainty, the development of user models based on training and performance experiments, and the adequacy of various presentation and input formats.

Acknowledgements

As with all systems developed in a co-operative environment, RUBRIC has benefitted from extended discussions with other members of the AI&DS technical staff. We would like to acknowledge recent contributions from Sonia Schwartzberg, Dan Shapiro, Brian McCune and Gerry Wilson.

References

(Dubois and Prade, 1982)
H. Prade, D.Dubois. A Class of Fuzzy Measures Based on Triangular Norms. *Int. J. General Systems*, 8:43-61, 1982.

(Lebowitz, 1983)
M. Lebowitz. Intelligent Information Systems. *Proc. 6th Int. ACM-SIGIR Conf. on R&D in Information Retrieval.* Bethesda, MD., 1983.

(Lukasiewicz, 1930)
J. Lukasiewicz. Many-valued Systems of Propositional Logic. In S. McCall, *Polish Logic*, O.U.P, 1957.

(Rescher, 1969)
N. Rescher. *Many Valued Logic*, McGraw-Hill, New York, 1969.

(Salton, 1971)
G. Salton (ed.). *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice-Hall Inc., Englewood Cliffs, NJ, 1971.

(Tong and Shapiro, 1984)
R.M. Tong, D.G. Shapiro. Experimental Investigations of Uncertainty in a Rule-Based System for Information Retrieval. *Int. J. Man-Machine Studies*. 1984 (to appear).

(Wilson *et al.*, 1983)
G.A. Wilson, E.A. Domeshek, E.L. Drascher, J.S. Dean. The Multipurpose Presentation System. *Proc. 9th Int. Conf. on Very Large Data Bases*. Florence, Italy, November 1983.

(Zadeh, 1965)
L.A. Zadeh. Fuzzy Sets. *Information and Control*, 8:338-353, 1965.

(Zadeh, 1973)
L.A.Zadeh. Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. *IEEE Trans. Systems, Man and Cybernetics*, SMC-3:28-44, 1973.